University Göttingen
GWDG/AG-C
Niklas Bölter, Hendrik Nolte

Exercise 1 / October 08, 2024
HPC System Administration / WiSe 202{4,5}
60 Minutes Total

Exercise Introduction

Before attempting the exercises in this document please ensure that you have read and understood the key topics covered in the tutorial.

Contents

Task 1: Understand the Overarching Goal (10 min)

As discussed in the lecture, we want to replicate a simple Beowulf cluster in a Openstack environment. Unfortunately we do not have time during the course to fully setup PXE Booting with warewulf. (**Note**: You can however try to setup your own private network with compute nodes after the workshop on your own. There are some hints in the optional exercises.)

The following figure shows the setup that will be used in the demo, where you can see that there are five virtual machines. One is the Warewulf management node (also called cluster manager), the other four nodes are compute nodes, managed by the Warewulf management node. The cluster manager has two network interfaces, one to the "Internet" and one to an internal network. The compute nodes have each only a single network interface to the internal network. If you follow the optional homework exercises and go to `Network -> Network Topology`, it should look similar to what is shown in Figure 1.

We actually have three different networks:

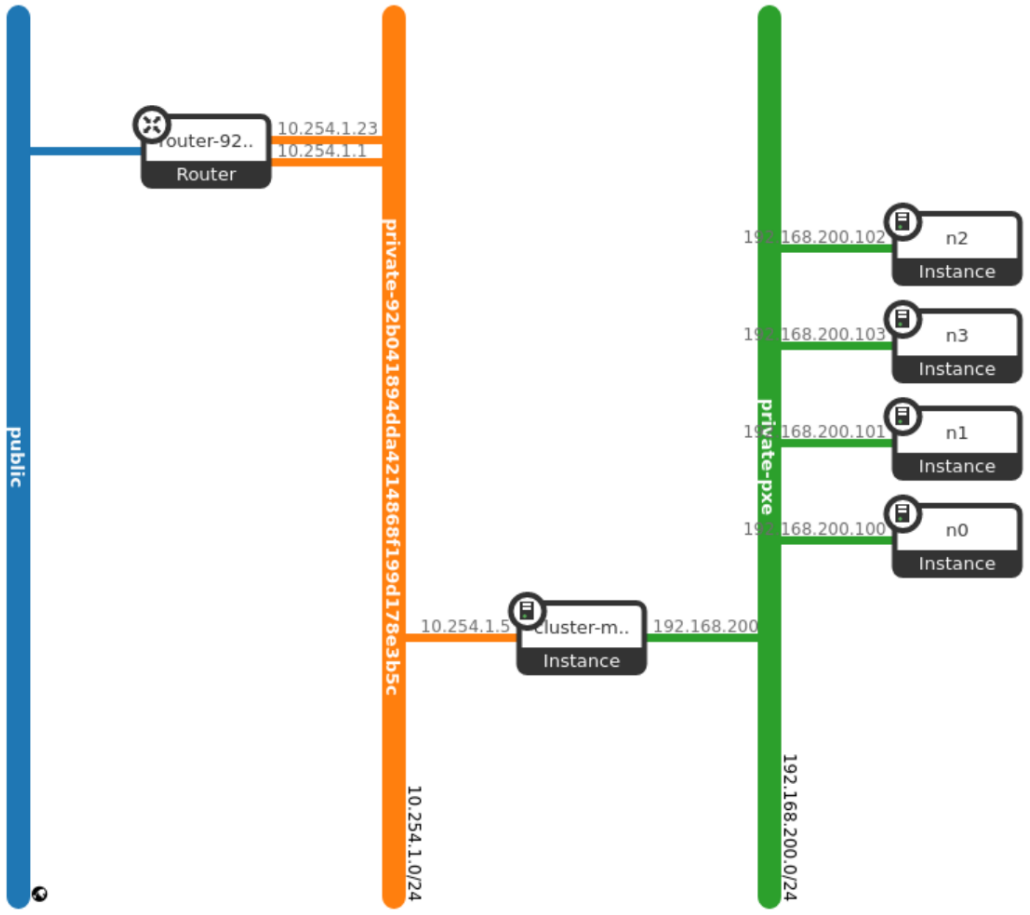

Figure 1: Network topology of our small cluster

- The left, public (blue) network is the internet, where you are coming from
- The internet is connected to a private (orange) network in the middle via a dedicated router. The cluster manager (Warewulf node) is connected to the orange network with the IP address 10.254.1.5
- The cluster manager can communicate to the internet using the router as a gateway. You can check this with:
  `$ ip route`
  Please keep in mind that in your case the IP addresses might be different.
- In addition there is another private network. It is located on the right (green). The cluster manager is connected to it too, and also the compute nodes are connected to this private network. Notice that it is easy to map between the internal IP adress and the hostname of the compute node. This is very useful when you have a large number of nodes!

The overarching goal of this tutorial is to configure cluster manager such that the compute nodes can do a PXE boot over the private network in the middle. That means, that instead of booting from disk, the compute nodes "boot from a network card". For this, they will

1. Configure their network after powering the node on. For this they make a dhcp broadcast which the dhcp server on the cluster manager will answer. This will configure their network and also provide the settings for the next step.
2. Download the iPXE stack via tftp from the cluster manager, once their network is configured.
3. Download the kernel, container and system overlay via http from the cluster manager using the iPXE

stack.

4. Execute the kernel and unpack the container and system overlay to provide a root file system.
5. Execute the containers `init` to finalize the bootstrapping of the image and start the operating system.

Task 2: Install Warewulf (10 min)

We are going to install Warewulf from the package manager. In order to omit the `sudo` in front of all the following commands, we log in as root. Since the user `cloud` has passwordless `sudo` configured this is easily doable with:
`$ sudo -i`
Check that your shell prompt has now switched from a user shell (`$`) to a root shell (`#`).
Before you install any new packages, make sure the system is fully updated by running `dnf update`. If foundational packages like kernel, systemd or glibc got updated, you should reboot the node to make sure all running processes are using the latest versions.
Now find the Warewulf User Guide (you can do this even if the cluster manager is rebooting!) and install the latest version of Warewulf using a Binary RPM according to the documentation [1].
If you check the output of your `dnf install` command carefully, you will see that a `dhcp-server`, a `tftp-server` and `nfs-utils` were installed along with Warewulf (some of them might already have been installed so they don't show up. You can use `rpm --query --requires warewulf` to see the full list of packages required by warewulf.) Warewulf needs the first two servers for its services and uses nfs exports to the compute nodes so that the user home directories on all nodes stay synchronized (see later).

Note: Confirmation dialogues on the command line look like this: `Yes means No and No means Yes. Delete all files [y/N]?` To answer the question you type `y` for **Yes** or `N` for **No** and hit enter.

Optional Task 3: Homework: Have a Look at the Network Configuration (5 min)

This is a difficult **additional** task which will support your understanding in the topic.

Note: Since the PXE Network Boot environment is not configured for this semester's course, you need to setup the network yourself after the course before you can do this exercise.

Before we continue with the configuration of warewulf, you can compare the network topology in the Openstack Dashboard (Horizon) with the configuration of the node. You can check the network interface with `ip addr` and the routing table with `ip route`.
`$ ip addr`
should show you two interfaces: `eth0` and `eth1`. You can compare the `inet` ipv4 addresses with your two private networks. If you look at your routing tables with
`$ ip route`
you can see when which interface and gateway is used to reach a certain ip address. Please check whether what you see is what you expected.

Task 4: Basic Configuration of Warewulf-Server (20 min)

All configuration files are located in `/etc/warewulf`. The warewulf server itself is configured within the `warewulf.conf`. You can go to the directory and then have a look at the file with:
`$ cd /etc/warewulf`

[1] We tested the exercise with version 4.5.7-1.

`$ cat warewulf.conf`
You can see, that it has a few general network configurations, and then four specific parts for the warewulf server, the dhcp server, the tftp server, and the nfs export. In order to enable PXE boot, we need to configure the network and the dhcp server.

Note: Since there is no second network this semester, you can just assign a second IP to the existing interface using `ip address add 192.168.200.1/24 dev eth0` to emulate it. This will allow warewulf to start up **using the configuration shown below** and listen on the interface. It will not actually do anything useful on the network since this is just a dummy IP adress which will have all packets silently dropped by OpenStack.

```
1  # Warewulf 4.5.7 default configuration
2  ipaddr: 10.0.0.1
3  netmask: 255.255.252.0
4  network: 10.0.0.0
5  dhcp:
6    enabled: true
7    template: default
8    range start: 10.0.1.1
9    range end: 10.0.1.255
10   systemd name: dhcpd
```

Please note that the line numbers do not match the ones in your actual config file. In order for warewulf to work properly you need to configure:

- `ipaddr:` This is the ip address of the cluster manager node itself. However, we have seen before with `ip addr`, that we have two network interfaces with two different ip addresses. Here you have to provide the ip address of the interface to the private-PXE network, where also the compute nodes are connected to. This as the network on the right in Figure 1.
- `netmask:` This is the netmask for the subnet of the private network, where later on, the entire cluster will be located. This is the corresponding network to the ip address you have chosen above. If you look closely in Figure 1 you can see that in this case it was written in CIDR notation as a `/24` network. This CIDR notation corresponds to a `255.255.255.0` netmask.
  Bonus Question: What network does the default warewulf config assume in CIDR notation?
- `network:` The private network for the PXE boot. If you look closely in Figure 1 you can see that in this case it was `192.168.200.0`.
- `range start:` First IP address to specify the range which should use this dhcp server. This ip address has to exist in the previously defined network.
- `range end:` Last IP address to specify the range which should use this dhcp server. This ip address has to exist in the previously defined network.

For more information about the other attributes you can have a look at the **Configuration** section of the Warewulf user guide.
Once you have made the necessary changes you can apply those using:
`$ wwctl configure --all`
This command will automatically configure all involved services. If you like to check this you can look into

- `/etc/dhcp` (and here maybe in `dhcpd.conf`) for the dhcp daemon config
- `/etc/hosts` to see the host configuration
- `/etc/exports` to see NFS exports (more on this in later sessions)

Warning: You should not alter any of these files if you don't exactly know what you are doing. Please just look into them using:
`$ cat <filename>`
Once you are satisfied you enable the warewulf server with:
`$ systemctl enable --now warewulfd`

The `--now` flag will also start the systemd unit after it has been enabled. And verify that the server is running with:

```
$ wwctl server status
```

Hints

- With reference to Figure 1, the `ipaddr` of the interface to the private-PXE network would be `192.168.200.1`.
- The `range start` and `range end` values refer to all the values that the `DHCP` can use. Since the X.X.X.1 adress is used by the cluster manager and nodes start at X.X.X.100, you should pick a range that is between these two in order to avoid conflicts between dynamic adresses handed out by DHCP and the statically configured ones. Bonus question: Why can the default configuration of warewulf use the full range from X.X.X.1 to X.X.X.255?

Task 5: Container Management (5 min)

The general purpose of Warewulf is to distribute operating system images to all nodes in a cluster, so that they are in sync with a golden image. This means that administrators only need to maintain a single image at a single source. In order to provide those images, Warewulf now supports Containers. However, please don't confuse these containers with the containers you know from your local Docker runtime. The container images are provisioned to bare metal without any virtualization layer on top. This is also the reason, why most Containers you find on Docker Hub, or other repositories, are not bootable in Warewulf. They are meant for a specific container runtime and usually lack a proper `systemd` to make them lighter. Therefore you should rather opt for container provided by warewulf [2].
The first step now is download a container image. You can download a rocky-8 Image with the following command:

```
$ wwctl container import docker://warewulf/rocky rocky-8
```

Please not that we have used the official Warewulf repository here. You can list all the available containers on your warewulf server (which requires you to have it downloaded beforehand) with:

```
$ wwctl container list
```

You should see an output listing our previously downloaded rocky-8 image.
This is a simple rocky-8 base image without much additional software installed. You can open a shell inside the container with the following command:

```
$ wwctl container shell rocky-8
```

Now use your shell to install some software. One example could be `nano`. Verify that it is not installed. You can use `dnf` to install some packages in this container in the same way like you do on the cluster manager VM itself. For instance you can run:

```
$ dnf install nano
```

Verify that now `nano` is installed on the system.
What happens when you exit the container depends on the exit code of the last command you ran. If the command was successfully (exit code 0), then you should see a "—write" in the prompt and exiting will rebuild the container image. You can experiment by running the `$ true` and `$ false` commands. Make sure your last command was successful and then leave the container shell with `exit`. Warewulf is now rebuilding and compressing the container image. The more packages you installed, the longer this will take.

Optional Task 6: Extension: Container Management (10 min)

This is a difficult **additional** task which will support your understanding in the topic.

[2] https://hub.docker.com/u/warewulf

I have mentioned in the lecture, that these containers are essentially uncompressed `chroot`. You can verify this statement by looking into `/var/lib/warewulf/chroots/`. here you find a `rootfs`. Verify that your `nano` was installed here.
If you check `/var/lib/warewulf/container/` you can see, that there exists also an container image.
What is the relationship between the `chroot` and the `image`?
How can you verify your assumption?

Task 7: Kernel Management (5 min)

You probably have seen this coming: In order to boot a Linux operating system, we need a Linux kernel to boot. There are generally two different ways to provide a Kernel. Some of the containers are shipped with a kernel. You can check this when looking at the output of
`$ wwctl container list --long`
Here you should see, that your rocky-8 container has a Kernel. This is particularly useful, if you are using a container with custom drivers (OFED, CUDA, etc.). However, you can also override the default kernel, or specify one, if no kernel was shipped with the container. For this you need to import a kernel. You can list all imported Kernels with:
`$ wwctl kernel list`
The output should be empty, since you did not import any new kernel until now.
So far, this was just for your information. We will have a closer look later.

Task 8: Adding a Node to Warewulf (10 min)

The configuration of the nodes is stored in the `nodes.conf` file (Remember: The config files are all located in `/etc/warewulf`). Navigate to the configuration directory and have a look at the file with:
`$ cat nodes.conf`
If you are starting from a clean installation, no nodes should be configured here. In order to configure a node, you can either edit the `nodes.conf` file with a text editor, or, and this is the recommended way (!), you can use `wwctl`, which then makes the appropriate changes to the `nodes.conf` file. In order to simply add a new node to Warewulf you can do:
`$ wwctl node add n0`
Afterwards you can check your configured nodes again with:
`$ wwctl node list`
In order to see all specified attributes of a node, please use:
`$ wwctl node list --all`
As you can see, for the newly added node there is no Kernel and no Container specified yet. In addition, the network of the node is also not specified. The minimum amount of attributes you need to configure are:

- `container` : The container name as shown by the `wwctl container list` command.
- `default:hwaddr` Hardware address (Mac-Address) of the network device
- `default:ipaddr` : The default assigned IP address.
- `default:netmask` the netmask of the network

you can also use
`$ wwctl node set --help` to see all options explained. We will now first configure the container image. You can set the container of a node with:
`$ wwctl node set --container rocky-8 n0`
You can check that the value was correctly set with:
`$ wwctl node list --all`
Next we need to configure the network, for which access to the OpenStack Dashboard (Horizon) is required.

Here, you have to go to `Compute -> Instances` and click on the instance name you want to add, e.g. `worker1` or `worker2`. You should see something similar to Figure 2. Please set the `hwaddr` to the shown MAC Address

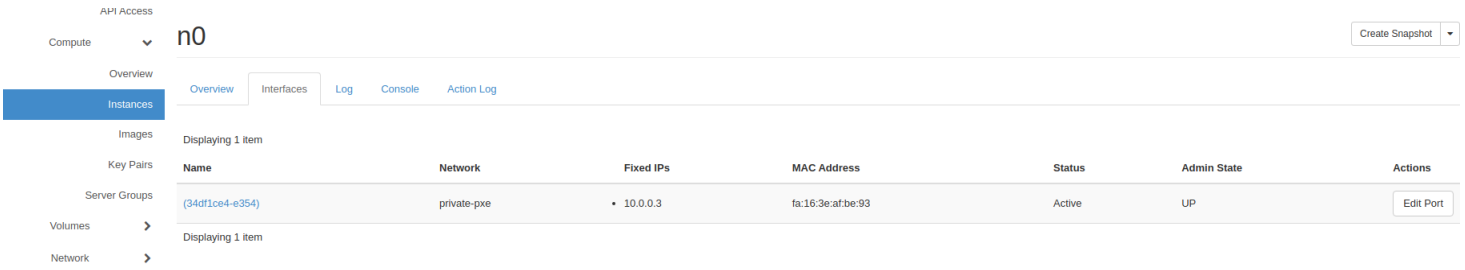

Figure 2: All required network information of the node n0 in Horizon. You can get there by clicking on `Compute` on the left and then on `Instances`. **Warning: Example output. Don't use these values.**

value with:
`$ wwctl node set --hwaddr fa:16:3e:af:be:93 n0`
Next we set the ip address, which is also shown with: `$ wwctl node set --ipaddr 10.0.0.3 n0`
The netmask of the private net of the node has been dicussed before. For simplicity we are going to use the same network for the PXE boot itself and for node operation afterwards. Therefore the netmask stays the same and you can set it using:
`$ wwctl node set --netmask 255.255.255.0 n0`
If you want to be sure, that every change is applied, you just force a rebuild of all configurations:
`$ wwctl configure --all`

Optional Task 9: Homework: Boot Your First Compute Node (60 min)

This is a difficult **additional** task which will support your understanding in the topic.

Note: Since the PXE Boot environment is not configured already for this semester's course, this task will take longer than 10 minutes to do at home. Some hints for setting this up: You need to create an internal network that is connected to the second interface of the cluster manager and all the compute nodes. If you do this with virtualbox on your own computer, disable UEFI boot, set the boot order to optical only and boot from an ipxe ISO like https://boot.ipxe.org/ipxe.iso using the virtio-net adapter. If you have access to the GWDG OpenStack cloud with your regular account, you might need to contact cloud-support@gwdg.de to increase your network and subnet quota from 1 to 2, create a new private network and subnet, add interfaces on the private network to all the VMs, disable port security on all the ports of the internal network (**BUT NOT ON ANY INTERFACE THAT IS REACHABLE FROM THE INTERNET VIA A FLOATING IP!**) and upload the ipxe ISO linked above as an image for the compute nodes to boot from. You do not necessarily have to configure any storage for the compute nodes, but they should have at least 8 GiB of memory for the boot to work (even more if you are booting a very large container). PXE Booting with VMs is very unreliable. You might have to restart a few times before it works. Check the console on the openstack website and then DHCP daemon and warewulf logs for problems.

Now that everything should be correctly configured for the first node *n0*, we should try to boot it. Since it might even be the case that the node was in a boot-loop before, it could be that the node already is booted up. You can try this by connecting to the node from your Warewulf node (Cluster Manager) via `ssh`:
`$ ssh n0`
Why this works, is explained later. If this worked and you are on the node *n0* (you can also check with the command `hostname`), then congratulations, you have successfully booted a node via PXE. If this doesn't work you can first check the output of the node in Horizon. For this you can click on `Console` which you can see in Figure 2. If there was some hick-up you can force a reboot with the `CtrlAltDel` button the top right of the console. You can also see this in Figure 3 If it didn't work, then the typical Job as a system administrator begins and we need to debug it. For this you can use the input on the console and the error log of warewulf

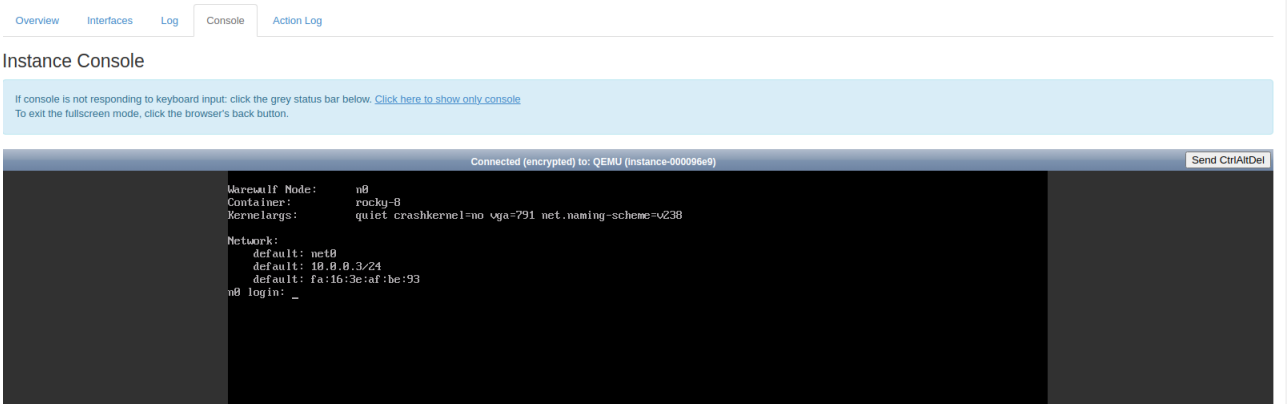

Figure 3: Here you see the console of the compute node available in Horizon. Please note the `CtrlAltDel` button in the top right corner.

which is located on the warewulf node in `/var/log`. You can see the last 30 messages in there for instance via:
`$ tail -n 30 /var/log/warewulfd.log`

Optional Task 10: Specifying a Custom Profile (10 min)

This is a difficult **additional** task which will support your understanding in the topic.

Instead of configuring each node individually, which would be extremely time consuming for a large, and probably even a small, cluster, one can group configurations in profiles. If you assign a node to a certain profile, the nodes attributes will inherit the ones from the profile. Every node which is added with `wwctl` will automatically be added to the `default` profile, as you can see when running:
`$ wwctl node list --all`
You can see if a certain value was inherited by a profile, that the profile name appears in `profile` column.
You can see all profiles with:
`$ wwctl profile list`
And similarly, you can check the values of each profile with:
`$ wwctl profile list --all`
You can create a new profile with:
`$ wwctl profile add test_profile`
And check that everything worked with:
`$ wwctl profile list`
You can set an attribute within this profile with:
`$ wwctl profile set --cluster cluster01 test_profile`
You can check all available attribute which you can set with:
`$ wwctl profile set --help`
And now go ahead and build your profile!
Once you are done, you can assign this profile to a node using:
`$ wwctl node set --profile test_profile n0`
YOu can now check again the values of the attributes of the node using
`$ wwctl node list --all`

Optional Task 11: Homework: Play with NFS Export (10 min)

This is a difficult **additional** task which will support your understanding in the topic.

You have seen during installation that nfs was installed and you have seen in the `warewulf.conf` that a nfs export was configured. Without any more commands, go please go ahead and

- create a `/data` directory and export it to the nodes
- insert some random data in it
- verify that you can read that on both compute nodes
- write on the compute to that export
- did it work immediately? Probably not. What was the problem?

This only scratches the surface of the `NFS` topic. There might be a session that goes into more depth later in the course.

Optional Task 12: Importing a New Kernel (20 min)

This is a difficult **additional** task which will support your understanding in the topic.

If you want to use a different kernel, you need to import one. We can for example import the kernel from Rocky Linux 8.8. This can be done with:

```
$ wwctl container import docker://rockylinux:8.8 rocky-8.8
$ wwctl container exec rocky-8.8 /usr/bin/yum install kernel
$ wwctl kernel import --container rocky-8.8 --detect
```

This command would now import the current kernel of the Warewulf node, including the kernel modules and firmware. You hopefully get an error message that tells you that the image of the kernel modules could not be created. A great learning opportunity!
Investigate the situation and fix the problem. (Hint: Can you convince the `wwctl` command to give you extra output? Are the inodes that create the problem important or can they be deleted?)
You can list all your already imported kernels using:

```
$ wwctl kernel list
```

If you want to see further what has been downloaded, you can have a look at `/var/lib/warewulf/kernel`. If you look closely during boot you can see that these are the files provided via http to the iPXE stack. Once a custom kernel has been imported one has to configure warewulf to use it, for instance to boot a specific node. You can do this for the *n0* node with:

```
$ wwctl node set --kerneloverride rocky-8.8 n0
```

You can, and should, check that everything was set correctly using:

```
$ wwctl node list --all n0 | grep Kernel.Override
```

Note:If you did the homework exercises above, you can now reboot the node. Afterwards you can ssh to the node and check the kernel:

```
$ ssh n0 'uname -r'
```