Exercise: Building a Warewulf Cluster

In this exercise, you will use your three provided Cloud-VMs to build your Compute Cluster with Warewulf on them. For this exercise, you need access to your Cloud-VMs, namely, you need to be able to log in, with SSH, into the first node, called head node, of the cluster. **IF YOU CANNOT LOG IN TO THE HEAD NODE, SPEAK NOW!**
Within this exercise, we will install Warewulf on the head node, and make it boot the other two nodes using PXE boot. **Important**: Your two other Cloud-VMs do not need to exist for now. We will create them during this exercise. For our cluster installation, we will use the default settings as much as possible. If the exercise sheet says something needs to be changed, it needs to be changed, as the default values do not work.

Contents

Please make sure to read all instructions carefully ...

Terminology

- **Head Node**: The head node of your Warewulf Cluster is the node that the warewulf service runs on. It is also the only node that you can log in with SSH directly and that can access the internet.
- **Compute Node**: The compute nodes of your Warewulf Cluster are the two VMs that Warewulf boots. They run stateless and cannot directly access the internet.
- **PXE boot network**: The Network on which the compute nodes boot on. It should be called like `private-pxe-XXXXXXXXXXXXXXX` and should have the network range `10.0.0.0/24`.

Task 1: Preparing the installation (7 min)

Before we begin, we'll do some basic preparation of the Head Node, to make it easier to work with.

1. Log into the Head Node with SSH and become `root` using `sudo -i`. You can stay `root` on the Head Node for the majority of this exercise.
2. Update the packages on the head node to the latest: `dnf update`.
3. Activate the "Extra Packages for Enterprise Linux" (EPEL) Repository, which holds some of our dependencies: `dnf install epel-release`.
4. Install your favourite text editor. Mine is `vim`: `dnf install vim`
   If you do not know which text editor to use, `nano` is probably a good option: `dnf install nano`
5. Install the following packages `htop`, `git`, `bind-utils`
6. Create the directory `/nfs` on the head node.
7. Familiarize yourself with the two internal networks that your OpenStack Project has: The outgoing network (usually named as `private-XXXXXXXXXXXXXXX`) and the PXE boot network (named `private-pxe-XXXXXXXXXXXXXXX`). Write down which IP addresses your head node has in both networks.

Task 2: Install Squid (3 min)

Just like in a "real" cluster, your Compute Nodes will not have direct access to the internet, as your OpenStack Project only has one Floating IP. Instead, they have to share the internet connection from your Head Node, using a proxy service called "Squid Cache". You can read more about it here: `https://www.squid-cache.org/Intro/`. Let's install this service on the Head Node.

1. Install the Squid Proxy service: `dnf install squid`
2. Enable and start the `squid` service: `systemctl enable --now squid`
3. Use `systemctl status squid` to check if the service is running fine.

Now, each of your Compute Nodes can use the internet connection from your Head Node, if you use the following commands:

```
export http_proxy=<pxe-ip-address-of-your-head-node>:3128
export https_proxy=<pxe-ip-address-of-your-head-node>:3128
```

WARNING: The default Squid config in `/etc/squid/squid.conf` is really dangerous to use on a production cluster! It is safe enough on this VM in the test cluster you are building, but on a production cluster, you would need to edit it to make it safe.

Task 3: Install Warewulf (10 min)

Next, we install the Warewulf Service on the Head Node. Warewulf's GitHub Project fortunately provides us with RPM files, which we can directly install. This RPM contains everything that warewulf needs, and we can install it straight from GitHub.

1. Install the `dnf download` plugin: Run `dnf install 'dnf-command(download)'`
2. Download the RPM file for Warewulf (Version 4.6.4) from GitHub: Run `dnf download https://github.com/warewulf/warewulf/releases/download/v4.6.4/warewulf-4.6.4-1.el9.x86_64.rpm`
3. Check that the file you've just downloaded is correct and wasn't tampered with by comparing its checksum to what is should be. You can get the checksum of the file with: `sha256sum warewulf-4.6.4-1.el9.x86_64.rpm`. The checksum should match the one listed on this page: `https://github.com/warewulf/warewulf/releases/tag/v4.6.4`.
4. **IF** the checksums match, install the package with `dnf install warewulf-4.6.4-1.el9.x86_64.rpm`
5. You can remove the RPM file afterwards again: `rm warewulf-4.6.4-1.el9.x86_64.rpm`
6. Warewulf normally updates `/etc/hosts` when it is reconfigured to include the names and IP addresses of all the nodes. Unfortunately, OpenStack replaces this file every boot, which means `wwctl configure hostfile` must be run every boot. To fix this, make an override for the `warewulfd` service file by creating the file `/etc/systemd/system/warewulfd.service.d/override.conf` and adding the following lines:

   ```
   [Service]
   ExecStartPre=/usr/bin/wwctl configure hostfile
   ```

   which will cause `/etc/hosts` to be regenerated before the Warewulf server is started. To make this override take effect, run `systemctl daemon-reload`
7. For all changes to take effect, log out of the head node and log back in.
8. Warewulf no longer includes iPXE images. This is not an issue with the VMs right now since you will be setting the compute nodes to boot iPXE from OpenStack, but with actual hardware you will need to provide iPXE. You can build iPXE images yourself (see `https://warewulf.org/docs/v4.6.x/server/bootloaders.html#building-ipxe-locally`) or install them (usually old versions) via:

   ```
   dnf install ipxe-bootimgs-x86 ipxe-bootimgs-aarch64
   ```

Task 4: Initial Warewulf Configuration (5 min)

After Warewulf is now installed, we need to set up its initial configuration in the `/etc/warewulf/warewulf.conf` file. This file is `YAML` formatted.

1. Look for the key `ipaddr` and set it to the IP address of your head node, the one from the PXE boot network.
2. Set the key `netmask` to `255.255.255.0`.
3. Set the key `network` to the network base of PXE Boot network, usually `10.0.0.0`.
4. Under `dhcp`, set the `template` to `static`.
5. Also under `dhcp`, set `range start` **and** `range end` both to 10.0.0.1
   Why? We do not want to use the DHCP feature of automatically assigning free IP addresses, instead, we want full control over which node gets which IP address. Thus, we set the range of IP addresses that

DHCP can hand out to an empty range.

6. Under `nfs`:

   a) Delete all entries

   b) Create one new entry, like this:

   ```
   - path: /nfs
     export options: rw,sync,no_root_squash
   ```

   This will convert the `/nfs` folder, that we created earlier, into a cluster-wide shared folder, which can be accessed from all nodes. We will use this folder later on, to install software in it to make it available on the entire cluster.

7. Apply the new configuration by calling `wwctl configure --all`

Task 5: Import a container (5 min)

As you have learned, Warewulf's main task is to provide an operating system for the cluster's booting compute nodes. To do this, we first need an operating system. One of Warewulf's main advantages over other cluster managers is that it can use containers, such as OCI or Docker containers, as operating systems. This makes management much easier, provided the containers have a kernel and an init system installed. However, typical DockerHub containers do not have these, as they are not intended to be used as an OS. While it is fairly simple to add the necessary packages to a standard DockerHub container — depending on the container, it could be as easy as running the command `dnf install kernel systemd` — the Warewulf team fortunately provides preconfigured containers ready to use with Warewulf for the Rocky Linux distribution, which are hosted on GitHub. We will simply use these.

1. Import a Rockylinux 9 container from GitHub into Warewulf, call it `node-rocky-9` and convert it into a bootable OS:
   `wwctl image import --build docker://ghcr.io/warewulf/warewulf-rockylinux:9 node-rocky-9`

Task 6: Update a container (5 min)

The container you have just imported and turned into an OS image might be severely out of date. Furthermore, it is just a basic installation, lacking almost any program to use on it. In this task, we will learn how to modify a container after it has been downloaded. Warewulf provides a simple tool for that, which is `wwctl image shell`. With `wwctl image shell` you can get a live shell inside any container you have downloaded, to change settings or install packages. If you are familiar, `wwctl image shell` works similar to `docker run -it` or `chroot`.

1. Open a shell inside the newly downloaded container. Run `wwctl image shell node-rocky-9`
2. Inside the shell, update the packages. Run `dnf update`
3. Install the EPEL Repository. Run `dnf install epel-release`
4. Next, install some useful packages inside the image. Run `dnf install htop git vim tar bind-utils`
5. Press `Ctrl + D` to exit the shell again. Notice how warewulf automatically rebuilds the image as soon as you leave the shell. This might take a moment.

Since we don't yet have any nodes, we can leave it at that. **Remember**: If you have any nodes booted from those images, your changes to the image will only be applied after you have rebooted the nodes.

Task 7: Create a node profile (10 min)

In Warewulf, we can use "Node Profiles" to apply the same values to multiple nodes in parallel, without typing them for each, such as the name of the container the nodes boot or which network interface they try first. By default, Warewulf creates one profile already, called `default`, and we will use this profile for all the values that our two nodes have in common.
To read the values for the `default` profile, we can use the command `wwctl profile list --all default` and to modify values for the profile, we can use the `wwctl profile set default`.

1. Use `wwctl profile list --all default` to read all existing values for the `default` profile. Take a guess what they mean.
2. Read through `wwctl profile set --help` for all values that can be set for a profile.
3. Set our recently imported container as the image for the `default` profile:
   `wwctl profile set --image node-rocky-9 default`
4. Set the profiles runtime overlays to `hosts`, `ssh.authorized_keys` and `syncuser` (thus, appending the syncuser overlay. This overlay automatically synchronizes usernames and groups from our head node to the cluster nodes).
   `wwctl profile set --runtime-overlays "hosts,ssh.authorized_keys,syncuser" default`
5. To include our `/nfs` share in the cluster nodes, we need to modify the configuration file directly. Open the file `/etc/warewulf/nodes.conf`, find the `default` profile in `nodeprofiles` and, in it, the key `resources > fstab`. Delete all entries in this key and add the following:

   ```
   - file: /nfs
     mntops: _netdev,defaults,nofail,rw
     spec: 10.0.0.XXX:/nfs
     vfstype: nfs
   ```

   Replace `10.0.0.XXX` with the IP address of your head node in the PXE boot network.

Task 8: Add your compute nodes in OpenStack (10 min)

Next, we can finally create the two other compute nodes of the cluster. Log in to the OpenStack Cloud website, under `https://cloud.gwdg.de` and go to the `Instances` tab on the left.
Repeat the following **twice**, once per compute node VM.

1. Click on `Launch Instance` (top right).
2. Give the new VM a name in the field `Instance Name`, e.g. `node2`, on the page `Details`.
3. On the `Source` page, search for the entry `IPXE` in the list under "Available", and click on the up-pointing-arrow of it, all the way on the right. Leave everything else unchanged on this page.
4. On the `Flavor` page, select `c1.large`.
5. On the `Networks` page, select the PXE Boot network.
6. On the `Security Groups`, and remove the default security group.
7. You can leave all other settings unchanged, click `Launch instance`

After your two instances are created, we'll now need to know their IP addresses and MAC addresses in order to set them up in Warewulf. For each compute node:

1. Click on the Instance name on the `Instances` page to get to their overview page.

2. Switch to the tab `Interfaces`
3. Find the interface of the PXE boot network, write down the IP address and MAC address listed here.
4. Click on `Edit Port` under `Actions` on the far right, it might be buried in the dropdown menu.
5. Uncheck the `Port Security` setting, then click update.

Task 9: Configure your compute nodes (10 min)

Now that we have created the two compute node VMs, we have to configure them in warewulf, such that warewulf can boot them. Switch back to the SSH session on the head node, and use the command `wwctl node add <nodename>` to add the node.

1. Firstly, use the command `wwctl node add --help` and read up on all the flags that this command allows.
2. Since we use a node profile, we do not need to set many settings for the individual nodes, they will inherit them from the profile. Use the command `wwctl node add --profile default --netname lan0 --netmask 255.255.255.0 --hwaddr XX:XX:XX:XX:XX:XX --ipaddr 10.0.0.XXX <nodename>` once per compute node VM to add it. Replace XX:XX:XX:XX:XX:XX with written down the MAC address of the node, and 10.0.0.XXX with the written down IP address of the node.
3. Use `wwctl node list --all <nodename>` to list all values for the node and make sure they are correct.
4. Apply the changes by running `wwctl configure --all`.
5. While the overlays for the nodes are configured, so far, they don't yet exist. Build the overlays for all nodes in warewulf with the command `wwctl overlay build`.

Task 10: Boot your compute nodes (5 min)

Now, with your nodes configured, they might already be booting.

1. Go onto the website, click on the name of each compute node VM, and click on the `Console` tab and check if the node has booted.
2. If not, you can press the button `Send Ctrl+Alt+Del` on the console tab, or the button `Hard Reboot Instance` in the dropdown menu on the top right to trigger a reboot.
3. If your nodes don't boot, report to your Tutor.

Task 11: Check SSH access to your compute nodes (5 min)

After all nodes have booted, verify that you can log in into them. Warewulf has automatically deployed an SSH key onto them; thus, you can simply use:

1. `ssh <nodename>`

If it asks for a password, that means Warewulf's generated SSH key for `root` hasn't been made yet or the nodes haven't gotten the public key from the overlays yet. Look inside `/root/.ssh`. If you don't see `cluster` and `cluster.pub` or the public key isn't in `authorized_keys` yet, run `source /etc/profile` to generate them (Warewulf configures the head node so that user SSH keys are generated on login). Rebuild the overlays for the nodes with `wwctl overlay build`, and then wait about 1-2 min for the nodes to reload the runtime overlay.

After verifying that all nodes can be logged in, try out the warewulf SSH tool for multi-node ssh:

1. Use `wwctl ssh <nodename-1>,<nodename-2> uptime` to check how long each node is already running.
2. Check if the `/nfs` share is available on all nodes by running `wwctl ssh <nodename-1>,<nodename-2> findmnt` and check if `/nfs` is in both outputs.

Task 12: Create yourself a user on the cluster (5 min)

Lastly: It is not good practice to use root all the time. In this task, you will create yourself an unprivileged user and deploy it around the cluster.

1. First, create the folder `/nfs/home`. This will be the location of the cluster user's home directory, such that their homes are available on all cluster nodes.
2. Choose a username.
3. Use the tool `useradd` to create a user on the head node. The `syncuser` tool will automatically deploy this username around the cluster. Use `useradd -b /nfs/home -m -U <username>` to create a user.
4. Rebuild your overlays, then wait 1-2 min for the `syncuser` overlay (remember, the one we added in task "Create a node profile") to synchronize the users, then change into the new user with `sudo -i -u <username>` on the head node, to check if the user works.
5. Warewulf automatically generated an SSH key for your user when you logged in during the previous step, which will be cluster-wide accepted. Try out if the ssh key works by using `ssh <nodename>` on your head node (as your new user) and see if you can log in to the node as your new user.

Task 13: Install DNS server on the head node (10 min)

The nodes you have just set up run in a very isolated network (your PXE boot network) and do not have any access to the outside world. This is typical for compute clusters, you usually do not want internet access on those nodes, especially because, Warewulf deploys nodes *stateless*, thus anything you download on the node is gone whenever they reboot. Occasionally, however, internet access is necessary. To do so, we'll first have to set up DNS access, such that your compute nodes can resolve, e.g. `gwdg.de` to `134.76.9.48`. Since your nodes do not have access to the network, they also cannot reach any DNS servers. We'll need to set up our own.

1. Verify that DNS access on the compute nodes does not work. Run `dig gwdg.de` on a compute node. It should hang for a while and then print
   `;; connection timed out; no servers could be reached`
2. Since the compute nodes cannot reach any DNS server on the internet, we'll have to set up our own on the head node. Fortunately, since DNS is recursive, our DNS server does not need to do more than forward all requests to one of the publicly available DNS servers on the internet, and forward the replies back to the compute node.
   On the head node, install the `bind` package, which contains the DNS server `named`: Run `dnf install bind`
3. Open the `named` config file `/etc/named.conf` in your favourite text editor.
4. In the line containing `listen-on port 53`, add `10.0.0.XXX;` into the parenthesis, where `10.0.0.XXX` is replaced with the PXE boot network IP address of your head node. This instructs `named` to listen on the PXE boot network for DNS requests.
5. In the line containing `allow-query`, add `10.0.0.0/24;` into the parenthesis.
   This allows your compute nodes to make queries to the DNS service.
6. Below that line, append the line `allow-recursion { localhost; 10.0.0.0/24; };`

This allows your compute nodes to make recursive queries.

7. Below that line, append the line `forwarders { 134.76.33.21; 134.76.10.46; };`
   This will redirect all queries from your compute nodes to the two primary GWDG DNS servers.
8. Now, save the file and check that the configuration file is correct: Run `named-checkconf`
   If everything is correct, it should not print anything.
9. Start and enable the `named` service: Run `systemctl enable --now named`

Task 14: Configure the compute nodes to use the DNS servers (5 min)

Now that we have a DNS server installed, we'll need to configure the compute nodes to use it. Fortunately, Warewulf provides us with the `resolv` overlay, which sets up the `/etc/resolv.conf` file of the nodes and thus can set the DNS server for us.. We'll just need to tell it, where the DNS server is located.

1. Check the names of your compute nodes networks, by using `wwctl node list`. The names are located under the tab *NETWORK*. Usually, they should all be named the same, usually `lan0`. If the names of the interfaces differ between your nodes, you will need to do the following part twice, individually for each node, and replace `wwctl profile set` with `wwctl node set`.
2. Set the IP address of the DNS server as a network tag in the Warewulf profile. This tag will give the `resolv` overlay the information, where to set the node's DNS server to. Run `wwctl profile set --netname lan0 --nettagadd DNS1=10.0.0.XXX default` where you need to replace `lan0` with the name of your compute node's network and `10.0.0.XXX` with the PXE network IP address of your head node.
3. Rebuild the overlays and reboot your nodes.
4. Go into your compute nodes once they are back up, and check that the `resolv` has properly set the DNS server, by checking the file `/etc/resolv.conf`. It should contain the line `nameserver 10.0.0.XXX` with `10.0.0.XXX` being the IP address of the head node on the PXE network.
5. Check that your compute nodes can now resolve server names with `dig`. Run `dig gwdg.de`

Task 15: Check that your nodes can access the internet (5 min)

With the combination of DNS server and Squid web proxy, your compute nodes now have (restricted) access to the internet. Check if they can download files.

1. Go onto one of your nodes.
2. Set the required environment variables to tell your programs to use the head node as a web proxy. Hint: They were shown in task 2.
3. Ensure the variables are correctly set by inspecting all set environment variables using the command `env`
4. Test out your compute node's internet connection by getting yourself an inspirational quote: Run `curl https://zenquotes.io/api/random`

Task 16: Create an overlay so nodes can access the internet (10 min)

In the previous task, you set the environment variables for the proxy manually. To set these for all users when they login to the nodes in a way that survives reboots, you will create an overlay.

1. On the head node, create an appropriately named overlay with `wwctl overlay create <name>`. The overlay will be at `/var/lib/warewulf/overlays/<name>`.
2. Create the file `etc/profile.d/squidproxy.sh` in that overlay, either manually or with `wwctl overlay edit`, and add exports for the proxy environment variables.
3. Add the overlay to system or runtime overlay for the `default` profile, using either `wwctl profile set` or editing `/etc/warewulf/nodes.conf` manually.
4. Rebuild the overlays with `wwctl overlay build`.
5. Reboot the compute nodes.
6. Login to one of the compute nodes.
7. Test out your compute node's internet connection by getting yourself an inspirational quote: Run `curl https://zenquotes.io/api/random`

Task 17: (Optional) Play with the cluster (5 min)

You have reached the end of the exercise sheet. This task is purely optional, if you have some more time to kill.
In this task, you should play with your cluster, to really get to know its structure and concepts. The bullet points below are merely suggestions, you can try them in any order, do all, do none, or invent your own.

- Make an overlay that uses templating.
  So far you have only made overlays with static content. Warewulf overlays can be templated to make files whose contents depend on the node. Make an overlay for your nodes that makes a file that is different on each one. Look at `https://warewulf.org/docs/v4.6.x/overlays/templates.html` and the builtin overlays under `/usr/share/warewulf/overlays` for how to do it.

- Build yourself a cluster monitoring script.
  Develop a script or program that gathers monitoring data from all nodes in your cluster and displays them in a clearly arranged view. Example metrics include: Uptime, CPU utilization, RAM utilization, network utilization, disk usage, etc.

- Install OpenMPI and run an MPI program.
  Create yourself a software folder in the nfs share (e.g. `/nfs/sw`) and install OpenMPI into it. Then try to use `mpirun` via SSH to run an MPI program on your two compute nodes.

- Do proper user management.
  Creating users manually on the head node and using the `syncuser` overlay to synchronize the users across the cluster is simple and effective, but on a real-life cluster deployment, you might want something more capable. Try to set up an OpenLDAP server on your head node and install SSSD on all nodes across the cluster, to have a more capable user management. Have a look into synchronizing SSH public keys, shells, and other user flags around the cluster.

- Fix Squid's configuration to be safe.
  Squid's default configuration is not safe for a production cluster. Fix its configuration to make it safe considering what networks you actually want to provide the proxy to and where they should be able to reach, and where not.

- Do proper security.
  Your head node has root access to all compute nodes. On a real cluster, this would be pretty powerful. To properly secure it, you should always have an SSH key with a password and protect it with a second factor.
  If you have a Yubikey, have a look into SSH-SK keys and secure your admin node with them.
  Otherwise, configure your admin node SSH to be protected by Google Authenticator or another OTP solution.
  WARNING: Be careful, especially with Google Authenticator. It is quite possible you will lock yourself out of your head node, and have to start at the beginning of this exercise sheet again. Always keep at least one open session in another terminal while trying new login settings!

- Do some speed tests on your Storage
  Your nodes use the `/nfs` share as shared storage. Make some speed tests on it with a proper benchmarking software, like IO500 or ElBencho. Which numbers can you reach?

- Get your cluster into the Top500
  Have a look at the 500 most powerful supercomputers in the world at `top500.org`. Their power is measured using the HPL benchmark. Download the HPL benchmark and run it on your cluster. How does your cluster compare to the Top500?