20230254

WIL620

Student Number: 20230254

Student Year: 2 (2024)

Exit Level Outcome 3

CTUTRAINING.AC.ZA | 0861 100 395 | ENQUIRY@CTUTRAINING.CO.ZA

CTU training solutions

Private
Tertiary
Education
Provider

Declaration of Authenticity

A critical aspect of any assignment is *authenticity*. Because you are completing much of the work for the assignments *unsupervised*, the examiner must be convinced that it is all your work. For this reason, you must complete the *Declaration of Authenticity* provided in the study guide and have it counter-signed by your manager, mentor, or lecturer.

NB	The declaration of authenticity is a legal document, and if found that you have made a false declaration, then not only will your results be declared null and void, but you could also have criminal charges brought against you. It is not worth taking the risk!

Please complete the declaration of authenticity below for all assignments:

DECLARATION OF AUTHENTICITY

I _____WILLIE DE KLERK_________________________________ hereby

declare that the contents of this assignment are entirely my work except for the following documents: (List the documents and page numbers of work in this portfolio that were generated in a group)

Activity	Date
Exit Level Outcome 3	**2024/07/26**

Signature: _______________ Willie _______________ **Date:** ________**2024/07/26**_____

Company/Mentor Stamp

Contents

Introduction

For the completion of exit level outcome 3 (Identify, install, support and troubleshoot system and application software) I was tasked to implement network function virtualization on a fusion broadband SD-WAN edge node that has x86 hardware and support for virtualization. As for the virtual network function I will be installing open source OpenWRT, adding a router and firewall with limited system resource consumption. Before the practical deployment, I would like to start by reviewing some of the groundwork for terminologies such as virtualization and hypervisors.

Virtualization is achieved when hardware like a server is coordinated in such a way that the guest operating system can share the hardware without any guest operating (VM) system knowing that they are sharing hardware. (Golden, 2007)

Hypervisors allow for the creation of virtual machines and performs hardware coordination to allow multiple VMs to run concurrently. (Edgeworth, et al., 2019) Hypervisors can be placed into two main categories, namely type 1 and 2:

Type 1: Runs directly on the system hardware and is also known as bare metal. (Edgeworth, et al., 2019)

Type 2: Runs on a host OS. (Edgeworth, et al., 2019)

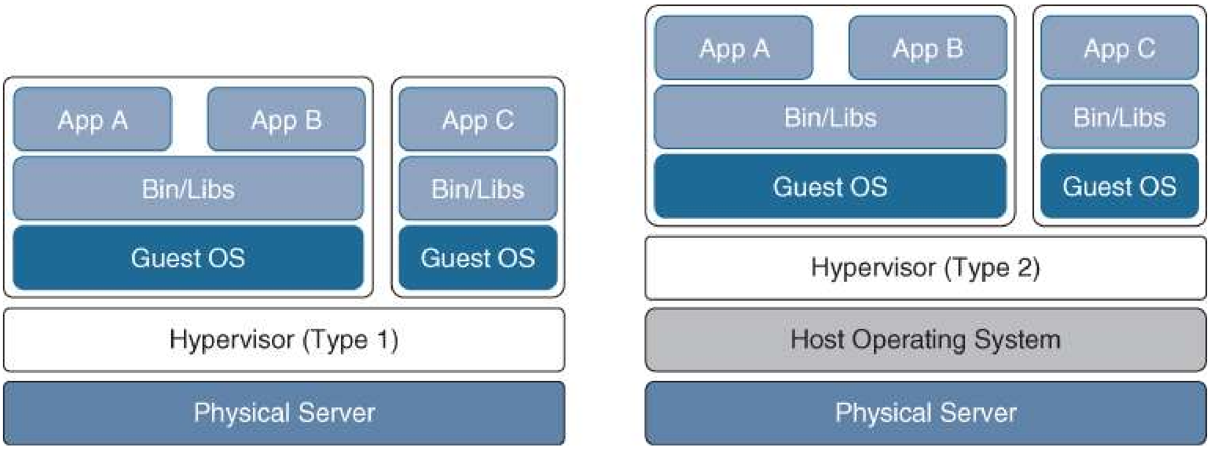


Figure 1 Hypervisor Types (Source: https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9780135262047/files/graphics/27fig02.jpg)

Virtualization has opened the door for the virtualization of network functions such as routing or firewalls though the Network Functions Virtualization (NFV) framework. This approach allows for reduced costs and space requirements along with the ability to test and deploy services without having to first develop hardware.

To achieve the task at hand a type 2 hypervisor will be used more specifically the open-source solution Kernel-based Virtual Machine (KVM) alongside QEMU and libvirt to provide an interface for management and provisioning.

To implement network function virtualization there are some requirements such as a processor supporting virtualization, and enough system resources that can maintain high load, still maintaining performance (storage, ram and processing cores) for the host operating system and the guest operating systems.

Checklist

Having a good checklist is always a good idea, thus I have made a small high-level checklist to aid with the implementation of network function virtualization.

Network Function Virtualization Checklist

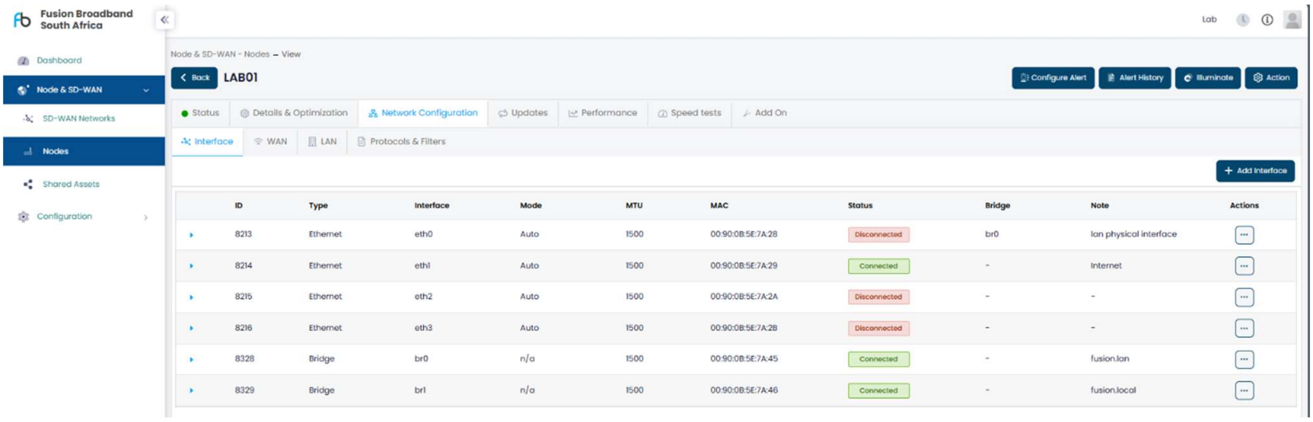
□ 1. Identify the LAN and Primary Internet interface on the physical device.
□ 2. Add two bridge interfaces to the device using the web-gui and bridge one port with the physical LAN interface.
□ 3. The second bridge interface should be assigned a IP address.
□ 4. Network Address Translation should be setup with the floating IP (public IP) being the source and the destination being the IP address assigned to the second bridge interface.
□ 5. Optionally DHCP can be implemented on the second bridge interface to allow OpenWRT to receive its 'wan' IP address via dchp.
□ 6. Libvirt should be installed on the edge node.
□ 7. The openwrt image should be downloaded
□ 8. The Virtual Machine should be created.
□ 9. Any initial configuration should be implemented through the console connection. (change root password and ssh port to 22222)
□ 10. Additional packages such as the Openwrt web gui should be installed.
□ 11. Openwrt should start automatically when the edge node starts up.

Implementation

Interface configuration

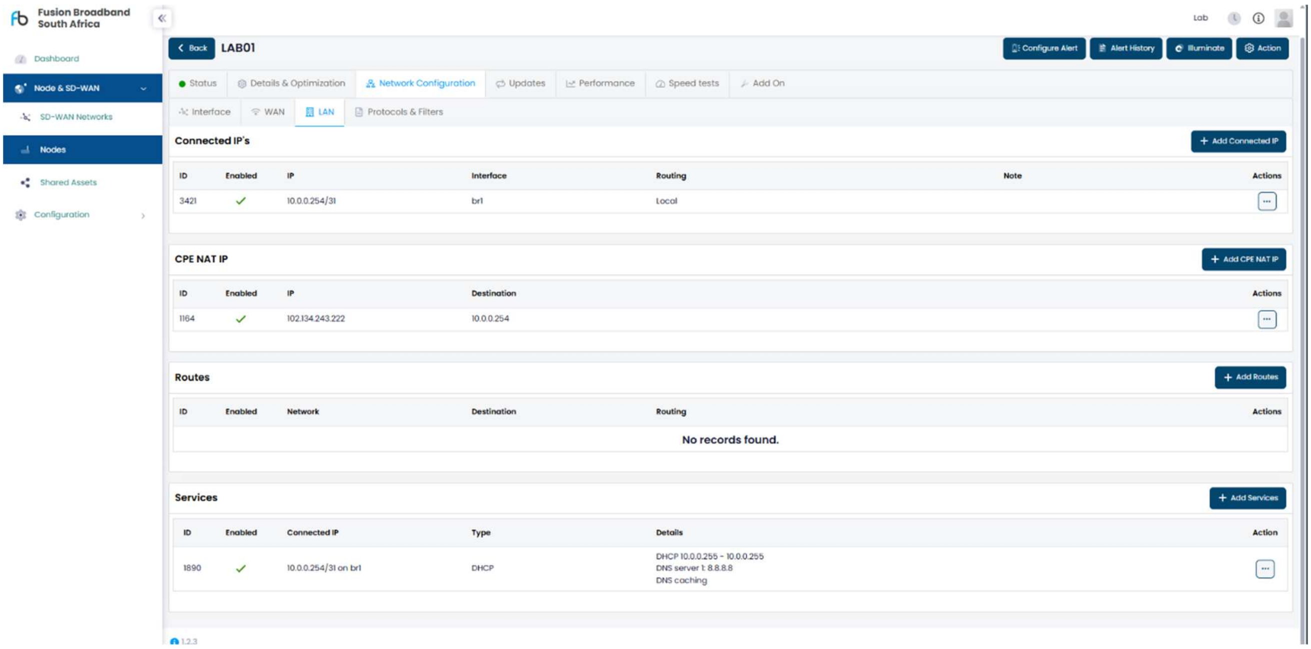
As per the first item on the checklist to identify the LAN and Primary internet Interface being used, I looked at the status lights on the node. With the interfaces being mapped clearly, I identified eth0 as the LAN interface and eth1 as the primary internet interface that is currently in use.

To add the bridge interfaces that to allow openwrt to communicate I logged in to the Fusion Broadband partner's exceptionally user-friendly web-gui using my credentials and two factor authentication code. Using the + add interface button I added two bridge type interfaces, giving it a note and assigning a MAC address. I then performed an action on eth0 to bridge with br0.



ID	Type	Interface	Mode	MTU	MAC	Status	Bridge	Note	Actions
8213	Ethernet	eth0	Auto	1500	00:90:0B:5E:7A:28	Disconnected	br0	lan physical interface	···
8214	Ethernet	eth1	Auto	1500	00:90:0B:5E:7A:29	Connected	-	Internet	···
8215	Ethernet	eth2	Auto	1500	00:90:0B:5E:7A:2A	Disconnected	-	-	···
8216	Ethernet	eth3	Auto	1500	00:90:0B:5E:7A:2B	Disconnected	-	-	···
8328	Bridge	br0	n/a	1500	00:90:0B:5E:7A:45	Connected	-	fusion.lan	···
8329	Bridge	br1	n/a	1500	00:90:0B:5E:7A:46	Connected	-	fusion.local	···

Next up I navigated to the LAN tab where I completed checklist items 3,4 and 5. A added an IP address to the second bridge interface, configured NAT and dhcp for the openwrt 'wan' interface.



Connected IP's

ID	Enabled	IP	Interface	Routing	Note	Actions
3421	✓	10.0.0.254/31	br1	Local		···

CPE NAT IP

ID	Enabled	IP	Destination	Actions
1164	✓	102.134.243.222	10.0.0.254	···

Routes

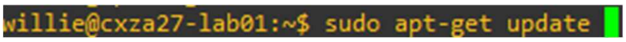
ID	Enabled	Network	Destination	Routing	Actions
No records found.					

Services

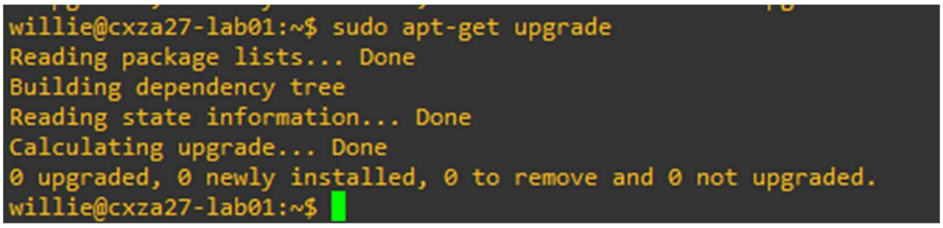
ID	Enabled	Connected IP	Type	Details	Action
1890	✓	10.0.0.254/31 on br1	DHCP	DHCP 10.0.0.255 - 10.0.0.255 DNS server 1: 8.8.8.8 DNS caching	···

Libvirt Installation

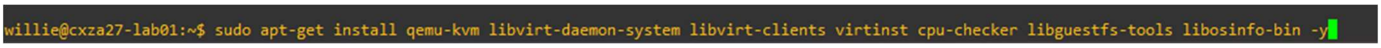
Taking 6 from the list I performed and apt update and upgrade, after which I installed libvirt and other required packages. I also needed to disable some default configurations.



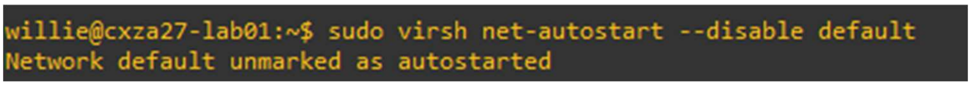
```
willie@cxza27-lab01:~$ sudo apt-get update
```



```
willie@cxza27-lab01:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
willie@cxza27-lab01:~$
```



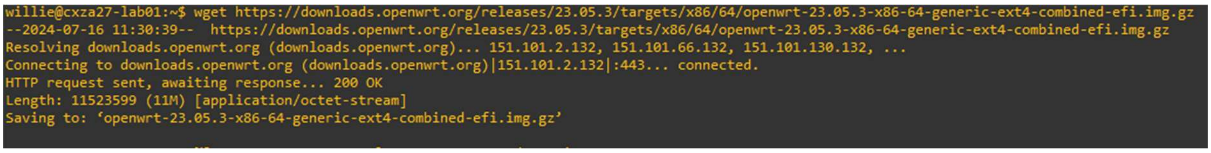
```
willie@cxza27-lab01:~$ sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-clients virtinst cpu-checker libguestfs-tools libosinfo-bin -y
```



```
willie@cxza27-lab01:~$ sudo virsh net-autostart --disable default
Network default unmarked as autostarted
```

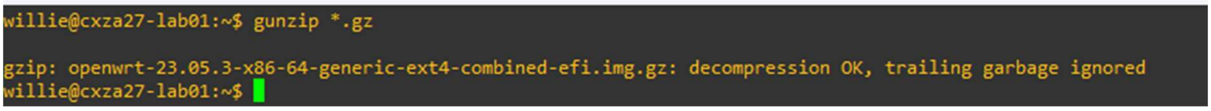
Openwrt VM Creation

To create the virtual machine I first needed to download the image.



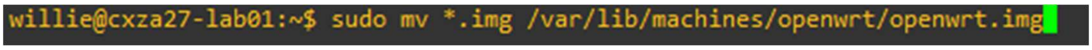
```
willie@cxza27-lab01:~$ wget https://downloads.openwrt.org/releases/23.05.3/targets/x86/64/openwrt-23.05.3-x86-64-generic-ext4-combined-efi.img.gz
--2024-07-16 11:30:39--  https://downloads.openwrt.org/releases/23.05.3/targets/x86/64/openwrt-23.05.3-x86-64-generic-ext4-combined-efi.img.gz
Resolving downloads.openwrt.org (downloads.openwrt.org)... 151.101.2.132, 151.101.66.132, 151.101.130.132, ...
Connecting to downloads.openwrt.org (downloads.openwrt.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11523599 (11M) [application/octet-stream]
Saving to: ‘openwrt-23.05.3-x86-64-generic-ext4-combined-efi.img.gz’
```

After downloading the image I had to unzip the image.



```
willie@cxza27-lab01:~$ gunzip *.gz

gzip: openwrt-23.05.3-x86-64-generic-ext4-combined-efi.img.gz: decompression OK, trailing garbage ignored
willie@cxza27-lab01:~$
```

I then moved the image to a created directory in /var/lib/machines/openwrt

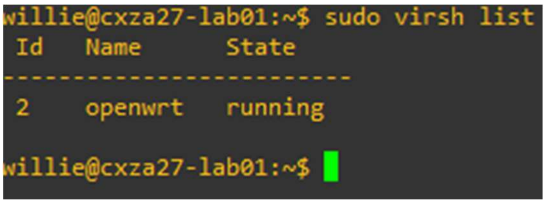


```
willie@cxza27-lab01:~$ sudo mv *.img /var/lib/machines/openwrt/openwrt.img
```

For the creation of the VM I used this command specifying the system resources, network interfaces and the path to the image that should be used.

sudo virt-install --os-type=generic --virt-type=kvm --name=openwrt --ram=512 --vcpus=2 --virt-type=kvm --hvm --network bridge=br0,model=virtio --network bridge=br1,model=virtio --connect qemu:///system --disk path=/var/lib/machines/openwrt/openwrt.img,bus=ide --import --wait 0

After creation I verified that it is running.

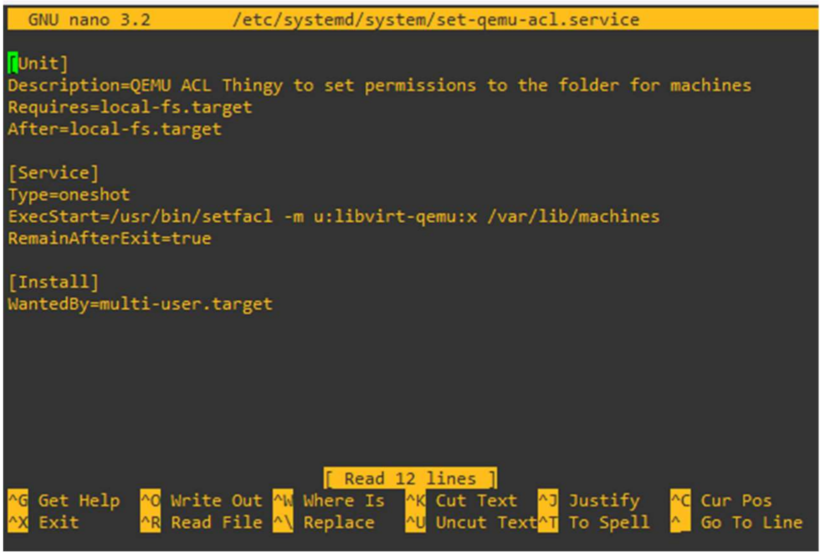


```
willie@cxza27-lab01:~$ sudo virsh list
 Id   Name      State
-------------------------
 2    openwrt   running

willie@cxza27-lab01:~$
```

Troubleshooting directory permissions

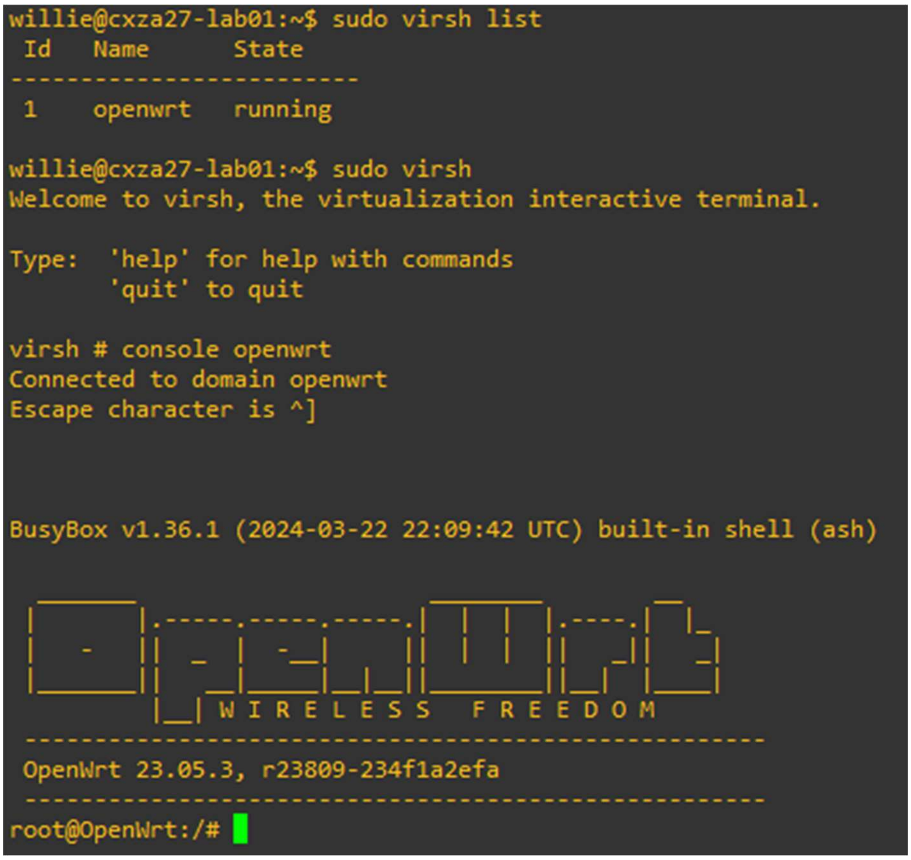
In my case there were issues with the libvirt-qemu user that made the /var/lib/machines directory non accessible. I solved this problem by the implementation of a startup service.

Implementing it as a service will prevent the permission changes from going away after a reboot.



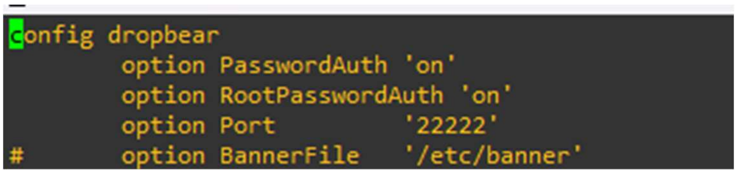
```
  GNU nano 3.2          /etc/systemd/system/set-qemu-acl.service

[Unit]
Description=QEMU ACL Thingy to set permissions to the folder for machines
Requires=local-fs.target
After=local-fs.target

[Service]
Type=oneshot
ExecStart=/usr/bin/setfacl -m u:libvirt-qemu:x /var/lib/machines
RemainAfterExit=true

[Install]
WantedBy=multi-user.target







                              [ Read 12 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

After this I used a console connection to the machine to allow me to make some basic changes.



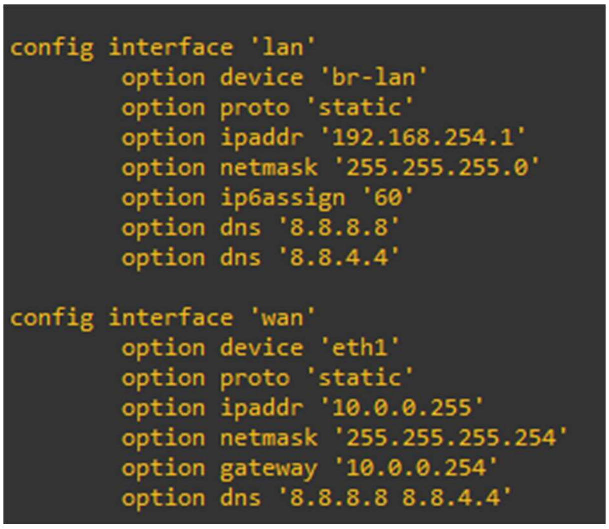
```
willie@cxza27-lab01:~$ sudo virsh list
 Id    Name       State
-------------------------
 1     openwrt    running

willie@cxza27-lab01:~$ sudo virsh
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # console openwrt
Connected to domain openwrt
Escape character is ^]



BusyBox v1.36.1 (2024-03-22 22:09:42 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 ---------------------------------------------------
 OpenWrt 23.05.3, r23809-234f1a2efa
 ---------------------------------------------------
root@OpenWrt:/#
```

First basic change that I made was add a root password and set the ssh port to 22222 in /etc/config/dropbear using vim editor. A quick important sidenote, to exit vim make use of the escape key and then press :wq to write quit alternatively :q to discard changes and just quit.



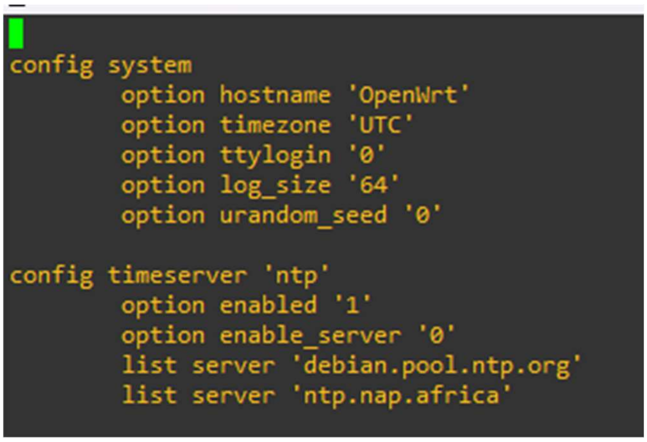
```
config dropbear
        option PasswordAuth 'on'
        option RootPasswordAuth 'on'
        option Port         '22222'
#       option BannerFile   '/etc/banner'
```

Within the network configuration file (/etc/config/network) , I changed the lan addressing and added static addressing for wan since I experienced technical difficulties with receiving addressing from dhcp. I used Google’s DNS servers instead of custom ones for testing purposes.



```
config interface 'lan'
        option device 'br-lan'
        option proto 'static'
        option ipaddr '192.168.254.1'
        option netmask '255.255.255.0'
        option ip6assign '60'
        option dns '8.8.8.8'
        option dns '8.8.4.4'

config interface 'wan'
        option device 'eth1'
        option proto 'static'
        option ipaddr '10.0.0.255'
        option netmask '255.255.255.254'
        option gateway '10.0.0.254'
        option dns '8.8.8.8 8.8.4.4'
```

I added two ntp servers to the openwrt system configuration found in /etc/config/system



```

config system
        option hostname 'OpenWrt'
        option timezone 'UTC'
        option ttylogin '0'
        option log_size '64'
        option urandom_seed '0'

config timeserver 'ntp'
        option enabled '1'
        option enable_server '0'
        list server 'debian.pool.ntp.org'
        list server 'ntp.nap.africa'
```

I also installed the argon theme using these commands:

wget https://github.com/jerrykuku/luci-theme-argon/releases/download/v2.3.1/luci-theme-argon_2.3.1_all.ipk -O $(basename https://github.com/jerrykuku/luci-theme-argon/releases/download/v2.3.1/luci-theme-argon_2.3.1_all.ipk) wget https://github.com/jerrykuku/luci-app-argon-config/releases/download/v0.9/luci-app-argon-config_0.9_all.ipk -O $(basename https://github.com/jerrykuku/luci-app-argon-config/releases/download/v0.9/luci-app-argon-config_0.9_all.ipk)

opkg update

opkg install luci-compat

opkg install luci-lib-ipkg

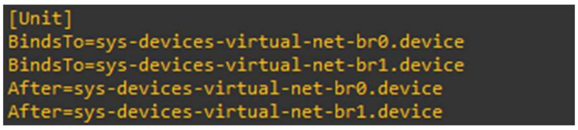
opkg install luci-theme-argon*.ipk

opkg install luci-app-argon-config*.ipk

opkg update && opkg install bash && wget https://raw.githubusercontent.com/dylanaraps/neofetch/master/neofetch && bash neofetch

Openwrt on startup

I added a few changes to the unit section of the libvirtd service to allow openwrt to have functioning interfaces on startup.



```
[Unit]
BindsTo=sys-devices-virtual-net-br0.device
BindsTo=sys-devices-virtual-net-br1.device
After=sys-devices-virtual-net-br0.device
After=sys-devices-virtual-net-br1.device
```

The following commands were used to enable the libvirt service on start-up, define and set the openwrt machine to start automatically.

sudo virsh define /etc/libvirt/qemu/openwrt.xml

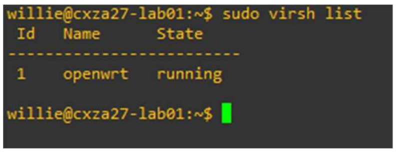
sudo virsh autostart openwrt

sudo apt-get install ebtables

sudo systemctl enable libvirtd

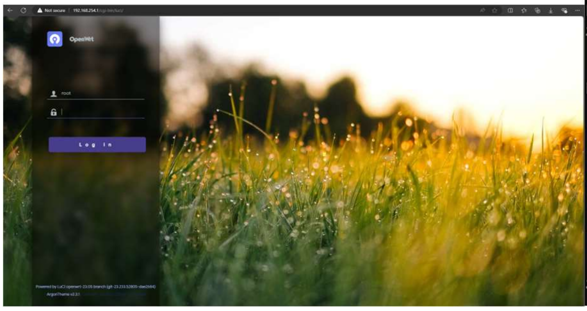
Verification

I am verifying that everything is working as I have set it up. Issuing the command below allowed me to view the operational status of the virtual machine created. I performed a reboot on the edge node and the vm started up as it should.



```
willie@cxza27-lab01:~$ sudo virsh list
 Id    Name       State
------------------------
 1     openwrt    running

willie@cxza27-lab01:~$
```

Verifying ssh access and web-gui access



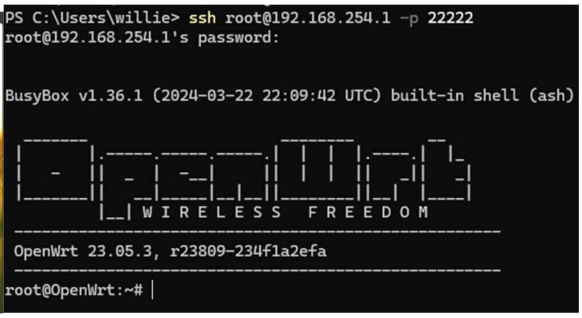
Not secure | 192.168.254.1/cgi-bin/luci/

OpenWrt

root

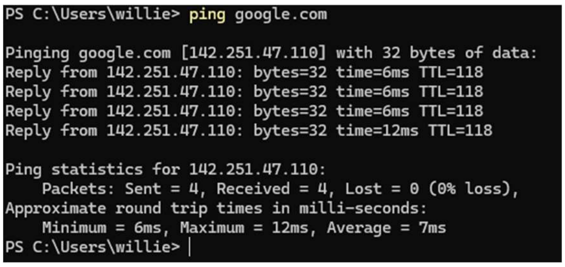
Log in

Powered by LuCI openwrt-23.05 branch (git-23.233.52805-dae2684)
ArgonTheme v2.3.1



```
PS C:\Users\willie> ssh root@192.168.254.1 -p 22222
root@192.168.254.1's password:


BusyBox v1.36.1 (2024-03-22 22:09:42 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 23.05.3, r23809-234f1a2efa
 -----------------------------------------------------
root@OpenWrt:~# |
```

Verifying internet connectivity



```
PS C:\Users\willie> ping google.com

Pinging google.com [142.251.47.110] with 32 bytes of data:
Reply from 142.251.47.110: bytes=32 time=6ms TTL=118
Reply from 142.251.47.110: bytes=32 time=6ms TTL=118
Reply from 142.251.47.110: bytes=32 time=6ms TTL=118
Reply from 142.251.47.110: bytes=32 time=12ms TTL=118

Ping statistics for 142.251.47.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 12ms, Average = 7ms
PS C:\Users\willie> |
```

Conclusion

During the task assigned to me for exit level outcome 3 I learned that the implementation of virtualization and network function virtualization allows for reduced costs and space requirements along with the ability to test and deploy services without having to first develop hardware. As an added use case it allows for the creation of lab environments with tools such as eve-ng and gns3.

My source for configuration: (Bartels, 2024)

Bibliography

Bartels, R., 2024. *hubandspoke.amastelek.com.* [Online]
Available at: https://hubandspoke.amastelek.com/deploying-a-fusionrt-edge-nfv-functionality-on-a-sd-wan-edge
[Accessed 16 07 2024].

Edgeworth, B., Rios, R. G., Gooley, J. & Hucaby, D., 2019. *CCNP and CCIE Enterprise Core ENCOR 350-401 Official Cert Guide | Chapter 27. Virtualization //Hypervisor Types.* [Online]
Available at: https://learning.oreilly.com/library/view/ccnp-and-ccie/9780135262047/ch27.xhtml#:-:text=Type%201%3A%20This,metal%E2%80%9D%20or%20%E2%80%9Cnative.%E2%80%9D
[Accessed 09 07 2024].

Edgeworth, B., Rios, R. G., Gooley, J. & Hucaby, D., 2019. *CCNP and CCIE Enterprise Core ENCOR 350-401 Official Cert Guide | Glossary //hypervisor.* [Online]
Available at: https://learning.oreilly.com/library/view/ccnp-and-ccie/9780135262047/gloss.xhtml#:-:text=hypervisor%20Virtualization%20software%20that%20creates%20VMs%20and%20performs%20the%20hardware%20abstraction%20that%20allows%20multiple%20VMs%20to%20ru

n%20concurren
[Accessed 09 07 2024].

Golden, B., 2007. *Virtualization For Dummies | Virtualization: A Definition.* [Online]
Available at: https://learning.oreilly.com/library/view/virtualization-for-dummies-r/9780470148310/ch01.html#:-:text=Virtualization%20refers%20to,anything%20at%20all.
[Accessed 09 07 2024].