*Systems Integration Assignment How-to*

*Configuring a server – C18490924*

1. *A DNS server for the domain example.lan, including both forward and reverse lookup.*

* First, we must install a DNS server. We will use a tool called BIND9, a tool for interacting with DNS. Run: **sudo apt install bind9**
* Next, we confirm it is installed by checking the version with: **named -v**
* We will use the name daemon, check if it is running first. Run: **systemctl status named**
* In the case it is not running we can use **sudo systemctl start named** to start the named daemon.
* Next, we will create a new zone, so we can use a DNS resolver to resolve example.lan to the IP address 10.20.30.40.
* Create a new zone file and configure it with the new example.lan records. To create the file run: **sudo nano /etc/bind/db.example.lan**
* In this file we will add the records to configure the zone file as below:

**$ORIGIN example.lan.**

**$TTL 123**

**@ IN SOA ns.example.lan. admin.example.lan. (**

**20211013**

**2d**

**15m**

**2w**

**1h**

**)**

**@ IN A 10.20.30.40**

**ns IN A 10.20.30.40**

**www IN CNAME example.lan.**

**@ IN NS ns.example.lan.**

* A handy trick to verify there is no errors in this file is by running the command: **named-checkzone example.lan. /etc/bind/db.example.lan**
* Next, we must make the name daemon aware of this new zone. To do this we open the named conf file with: **sudo nano /etc/bind/named.conf**
* We then add the following lines:

**zone "example.lan" in {**

**type master;**

**file "/etc/bind/db.example.lan";**

**};**

* We now must restart the name daemon to load these changes. Run: **sudo systemctl restart named**
* Verify it works by running **dig @127.0.0.1 example.lan** – you should see 10.20.30.40 in the answer.
* Next, we must set up the server for reverse lookup. First create a reverse lookup file in the same location as the example.lan zone file with: **sudo nano /etc/bind/40.30.20.10.in-addr.arpa**
* Add the following records to this file:

**$TTL 123**

**40.30.20.10.in-addr.arpa. IN SOA ns.example.lan. admin.example.lan. (**

**20211013**

**2d**

**15m**

**2w**

**1h**

**)**

**40.30.20.10.in-addr.arpa. IN PTR example.lan.**

**40.30.20.10.in-addr.arpa. IN NS ns.example.lan.**

* Similar to the example.lan zone we created we must make the name daemon aware of this new file. Once again run: **sudo nano /etc/bind/named.conf**
* Add the following record to this file below the example.lan zone:

**zone "40.30.20.10.in-addr.arpa" in {**

**type master;**

**file "/etc/bind/40.30.20.10.in-addr.arpa";**

**};**

* Once again, we must restart the name daemon to load these changes. Run: **sudo systemctl restart named**
* Finally, we should be able to get the reverse lookup (example.lan) of our ip address 10.20.30.40 with: **dig @127.0.0.1 -x 10.20.30.40**

1. *A DHCP server, which can lease clients an IP address in the range 192.168.1.150 to 192.168.1.200*

* Prior to starting this, we need both a server and a client connected via an internal network to lease the IP address to.
* First, we install the DHCP server on the server. Run: **sudo apt install isc-dhcp-server**
* Next, we must check which interface the client is connected to the server with. To do this we must run two commands and compare them. The interface not visible on the second command is the one we need. Run: **ip link** & **ifconfig**
* Next, we must tell the DHCP server which interface it is going to be leasing an ip address on. To do this we must access the config files of the dhcp server we installed first. Run: **sudo nano /etc/default/isc-dhcp-server**
* Find the line that says INTERFACESv4=”” and add the name of the interface we got previously in between the quotes.
* Next, we update the DHCP configuration file to let it know we want to lease IPs in the range 192.168.1.150 to 192.168.1.200. To do this we open the conf file with: **sudo nano /etc/dhcp/dhcpd.conf**
* Add the following to this file:

**subnet 192.168.1.0 netmask 255.255.255.0 {**

**range 192.168.1.150 192.168.1.200;**

**option subnet-mask 255.255.255.0;**

**option routers 192.168.1.1;**

**option broadcast-address 192.168.1.255;**

**default-lease-time 600;**

**max-lease-time 30000;**

**}**

* Next, we must update the netplan to make it aware of the interface we shall be using DHCP on. Open the 99\_config.yaml file with: **sudo nano /etc/netplan/99\_config.yaml**
* Then add the following record into this file (if your interface is different, it must be replaced):

**network:**

**version: 2**

**renderer: networkd**

**ethernets:**

**enp0s8:**

**addresses:**

**- 192.168.1.150/24**

**gateway4: 192.168.1.1**

* To give this server the static IP address and save these changes we must apply the new netplan. To do this run: **sudo netplan apply**
* Before moving onto the client, we restart the DHCP server with the command: **sudo systemctl restart isc-dhcp-server**
* On the client, we must once again find which interface to use by running the same commands as before: **ip link** & **ifconfig**
* Then we update the netplan like we did on the server but add a different record as this will be assigned through DHCP. Run: **sudo nano /etc/netplan/99\_config.yaml**
* Add the following to this file (changing the interface if needed):

**network:**

**version: 2**

**renderer: networkd**

**ethernets:**

**enp0s8:**

**dhcp4: true**

* Once again, we apply the netplan with: **sudo netplan apply**
* Finally, reboot the client and run **ifconfig**. You should see the ip address in the range 192.168.1.150 to 192.168.1.200

1. *An NFS server, allowing the file /home/<username>/shared on the server VM to be shared with clients*

* First, we must install the NFS server with: **sudo apt install nfs-kernel-server**
* Once this is running, we must create a folder we wish to share. To do this run: **mkdir shared** – or whatever you wish to name the folder
* We must now change the permissions on this folder to meet the needs of the NFS server. We do this by running the following two commands: **sudo chown -R nobody:nogroup shared/** & **sudo chmod 777 shared/**
* We now must let the NFS server know we want to share the directory we just created. Open its config file with: **sudo nano /etc/exports**
* Next add the following record to the file, replacing the username and ip as needed: **/home/stephen/shared 192.168.0.250(rw,async,no\_subtree\_check)**
* We must now update the server to these changes by running the following two commands: **sudo exportfs -a** & **sudo systemctl restart nfs-kernel-server**
* Finally on the server we must allow NFS connections through the firewall. Again, you must replace the ip as needed. Run: **sudo ufw allow from 192.168.1.152 to any port nfs**  & **sudo ufw enable**
* Over on the client we must install the NFS client software with: **sudo apt install nfs-common**
* Next, we must create another directory to mount the shared nfs directory too. This can be called anything. Run: **mkdir nfs\_dir**
* Finally, mount the shared directory with the following command, once again replacing the ip and username as needed: **sudo mount 192.168.1.150:/home/stephen/shared nfs\_dir** – this is now complete

1. *An FTP server, allowing files stored on the server to be transferred to a client*

* First, on the server we must install the FTP server with: **sudo apt install vsftpd**
* Next, we must modify its config file. Open it with: **sudo nano /etc/vsftpd.conf**
* Find the line that says anonymous\_enable and change it from **NO** to **YES** – this will allow anonymous users to connect to our ftp server.
* Next, we must tell the server which ports to use, we do this by adding the following lines: **pasv\_enable=YES** & **pasv\_min\_port=10000** & **pasv\_max\_port=10010**
* For this to be applied we must restart the server with: **sudo systemctl restart vsftpd**
* Finally, we must open the firewall to allow connections on the ports we set up earlier. We can do this with: **sudo ufw allow 21/tcp** & **sudo ufw allow 10000:10010/tcp**
* Now if we place a file in the FTP servers /srv/ftp/ directory we are able to get it using the wget command like so: **wget ftp://** **192.168.1.150 /my\_file.txt**

1. *A router, allowing a client machine connected to the server to access the Internet*

* Firstly, on the server we open the config file sysctl.conf to enable packet forwarding. To do this run: **sudo nano /etc/sysctl.conf**
* Next at the following line to allow ipv4 forwarding, or if its already there uncomment it: **net.ipv4.ip\_forward=1**
* Next, we must set up the server to forward packets from the enp0s8 interface to the enp0s3 interface. We do this with: **sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT**
* We now must do the same but in reverse so that packets can go from enp0s3 to enp0s8. Run: **sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -m state –state RELATED,ESTABLISHED -j ACCEPT**
* Next, we enable network address translation so that the servers unique ip can represent the client too. To do this we run: **sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE**
* In order to permanently save these, we must use something called iptables-save as you can not do this by default. To install this, we use: **sudo apt install iptables-persistent**
* Now we can finally save the changes we made by running the following: **sudo bash -c "iptables-save > /etc/iptables/rules.v4"**
* Next, we must change the firewall so that it allows NAT packets to be forwarded to our client. Open the config file used for this by running: **sudo nano /etc/default/ufw**
* To get our UFW to allow packet forwarding we must find the line in here that says **DEFAULT\_FORWARD\_POLICY="DROP"** and change **DROP** to **ACCEPT**.
* Next, we must set ip forwarding to be true in the /ufw/sysctl.conf/ file. To do this open the file with: **sudo nano /etc/ufw/sysctl.conf**.
* Then either add the following line or it may be commented in the file to set ip forwarding to true: **net/ipv4/ip\_forward=1**
* Last but not least we must restart the firewall and save the changes we have made. This can be done simply with the following three commands:

**sudo ufw disable** & **sudo ufw enable** & **sudo netfilter-persistent save**

* Now we switch over to the client where we must modify its network configuration to allow it to connect to the router successfully.
* The first network configuration file we must change is /etc/netplan/00-installerconfig.yaml. This is no longer used so either delete or comment out the contents. Open it with: **sudo nano /etc/netplan/00-installerconfig.yaml** and do either one.
* The next network config file to be changed is our 99\_config.yaml file. Open it with **sudo nano /etc/netplan/99\_config.yaml**
* Add the following record so it knows to use the router we set up on the server for its network – changing the ip addresses and interface where appropriate:

**network:**

**version: 2**

**renderer: networkd**

**ethernets:**

**enp0s8:**

**dhcp4: true**

**gateway4: 192.168.0.117**

**nameservers:**

**addresses:**

**- 192.168.0.117**

**- 192.168.0.1**

* To apply these changes to the client we must run the command **sudo netplan apply**
* Finally, if we disable our other network interface the client should still have a valid internet connection if successfully connected to our router!