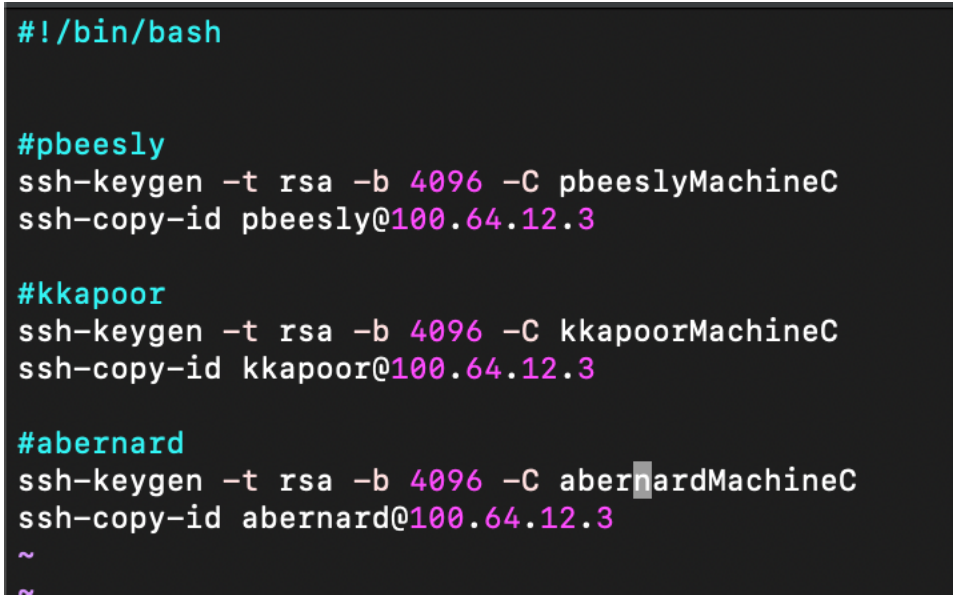
Remainder of the tasks:

Passwordless Logins:



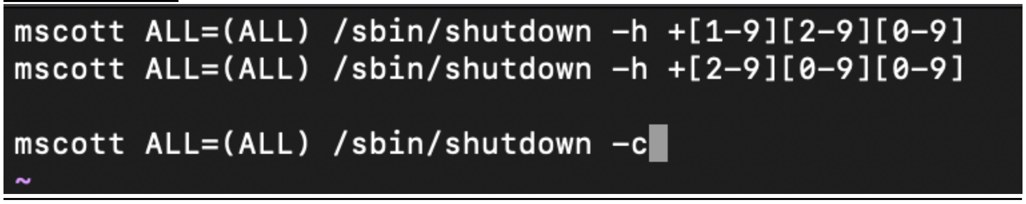
```
#!/bin/bash


#pbeesly
ssh-keygen -t rsa -b 4096 -C pbeeslyMachineC
ssh-copy-id pbeesly@100.64.12.3

#kkapoor
ssh-keygen -t rsa -b 4096 -C kkapoorMachineC
ssh-copy-id kkapoor@100.64.12.3

#abernard
ssh-keygen -t rsa -b 4096 -C abernardMachineC
ssh-copy-id abernard@100.64.12.3
~
~
```

This script allows for pbeesly, kkapoor, and abernard to ssh into machine C without a password. I simply use the ssh-keygen command to create the key, and create a note of whose key it is. I then do a ssh-copy-id command at the IP of machinec. I made the personal decision to log into each user listed above and run these commands manually. I have this 'script' for safe keeping remembering what I did.

DSchrute Sudo:

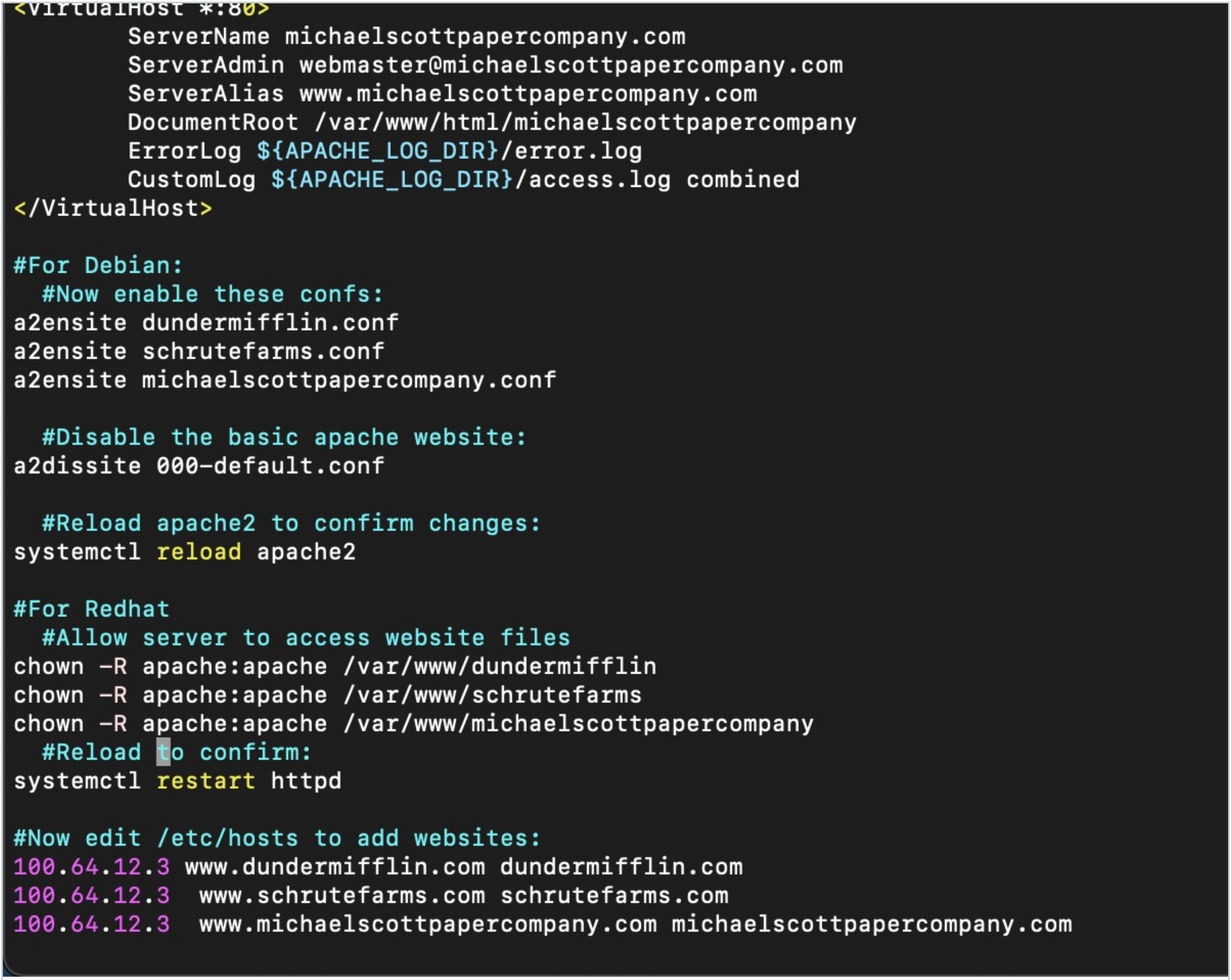
To allow Dwight to run any command via sudo I went into /etc/sudoers.d for each machine and created the file 'dschruteRoot'. It contains the command: dschrute ALL=(ALL:ALL) ALL. This allows for Dwight to basically run any command in sudo. I then used the scp command to copy it to the other machines.

Mscott Sudo:



```
mscott ALL=(ALL) /sbin/shutdown -h +[1-9][2-9][0-9]
mscott ALL=(ALL) /sbin/shutdown -h +[2-9][0-9][0-9]

mscott ALL=(ALL) /sbin/shutdown -c
~
```

To allow Michael to shutdown machines BCDF with a time limit in minutes of 120-999 minutes and also allow him to cancel the shutdown, I created the file 'mscottShutdown' in the /etc/sudoers.d directory. The file is shown above in the screenshot. It contains the user mscott is allowed to run the command shutdown located in sbin. Flag -h haults the machine and the +[number] constitutes how much time in minutes it will shut down from now. To limit 120-999, I used some regex to limit certain numbers as being a valid input for this command. The last line of the script allows for Michael to cancel the shutdown.



```
<VirtualHost *:80>
        ServerName michaelscottpapercompany.com
        ServerAdmin webmaster@michaelscottpapercompany.com
        ServerAlias www.michaelscottpapercompany.com
        DocumentRoot /var/www/html/michaelscottpapercompany
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

#For Debian:
  #Now enable these confs:
a2ensite dundermifflin.conf
a2ensite schrutefarms.conf
a2ensite michaelscottpapercompany.conf

  #Disable the basic apache website:
a2dissite 000-default.conf

  #Reload apache2 to confirm changes:
systemctl reload apache2

#For Redhat
  #Allow server to access website files
chown -R apache:apache /var/www/dundermifflin
chown -R apache:apache /var/www/schrutefarms
chown -R apache:apache /var/www/michaelscottpapercompany
  #Reload to confirm:
systemctl restart httpd

#Now edit /etc/hosts to add websites:
100.64.12.3 www.dundermifflin.com dundermifflin.com
100.64.12.3  www.schrutefarms.com schrutefarms.com
100.64.12.3  www.michaelscottpapercompany.com michaelscottpapercompany.com
```

This script above shows how I installed the web servers for the three websites on Debian and redhat. There are some basic notes on what the commands do, but here is my general thinking. I copied the unzipped .tgz files into the html directory and created conf files so the websites would work. I then enabled the websites and created a DNS like entry in /etc/hosts. The Debian vs redhat commands were a bit different, initially I had to chown permission to apache on redhat to allow the server to read all the files. Other than that it was pretty straight forward. This is not a direct script, but these are the commands I used to set the servers up and is used as a note reference file.

For the cron jobs on machine c, I used this command in the crontab file:

*/5 * * * * rsync -aH 100.64.12.3:/var/www/html/dundermifflin/* /var/www/html/dundermifflin

*/5 * * * * rsync -aH 100.64.12.3:/var/www/html/schrutefarms/* /var/www/html/schrutefarms

12 loch1722 5030 ChayetLogan
10/10 Apache running and enabled
20/20 Web Site Access

10/10 Web Site Alias
10/10 Users can create files
10/10 Apache can read user files
10/10 Group can modify files
10/10 Others cannot modify files
8/10 Backup web site running and enabled
3/10 Backup web site access
0/20 Extra credit
0/20 Bonus credit
Final Grade 91/100

Cannot get http://www.dundermifflin.com/images/DM-logo-222x120.jpg from web server D
Cannot get http://www.dundermifflin.com/kkapoor-mirror from web server D
Cannot get http://www.dundermifflin.com/abernard-mirror from web server D
Cannot get http://www.dundermifflin.com/pbeesly-mirror from web server D
Cannot get http://www.schrutefarms.com/dschrute-mirror from web server D
Cannot get diskusage on C
Cannot get diskusage on D

Willem Schreuder , Oct 3 at 8:53am

Add a Comment:

Media Comment **Attach File**

Save

To limit logins using the PAM module, I first enabled the /etc/security/access.conf on all machines by typing in the command: account required pam_access.so to the following files:

- Redhat
  - /etc/pam.d/system-auth
  - /etc/pam.d/password-auth
- Debian
  - /etc/pam.d/login
  - /etc/pam.d/sshd

Then I edited the /etc/security/access.conf according to specific conditions in the assignment:

- +:root loch1722 mscott dschrute : ALL → This allowed for these users to access the machine and was applied to all machines
- +: (All dunder Mifflin usernames) : ALL → This allowed for all the users at Dunder Mifflin to access machine E
- +:pbeesly abernard kkapoor : ALL → This allowed for these users to access machines C and D
- +:(accounting) : ALL → This allowed for users in the accounting group to access machine F
- -:ALL:ALL → This was put at the end of every access.conf file as to not allow access to anyone else but specified above

To set the password policy, I went into /etc/security/pwquality.conf and edited the file to add the policy:

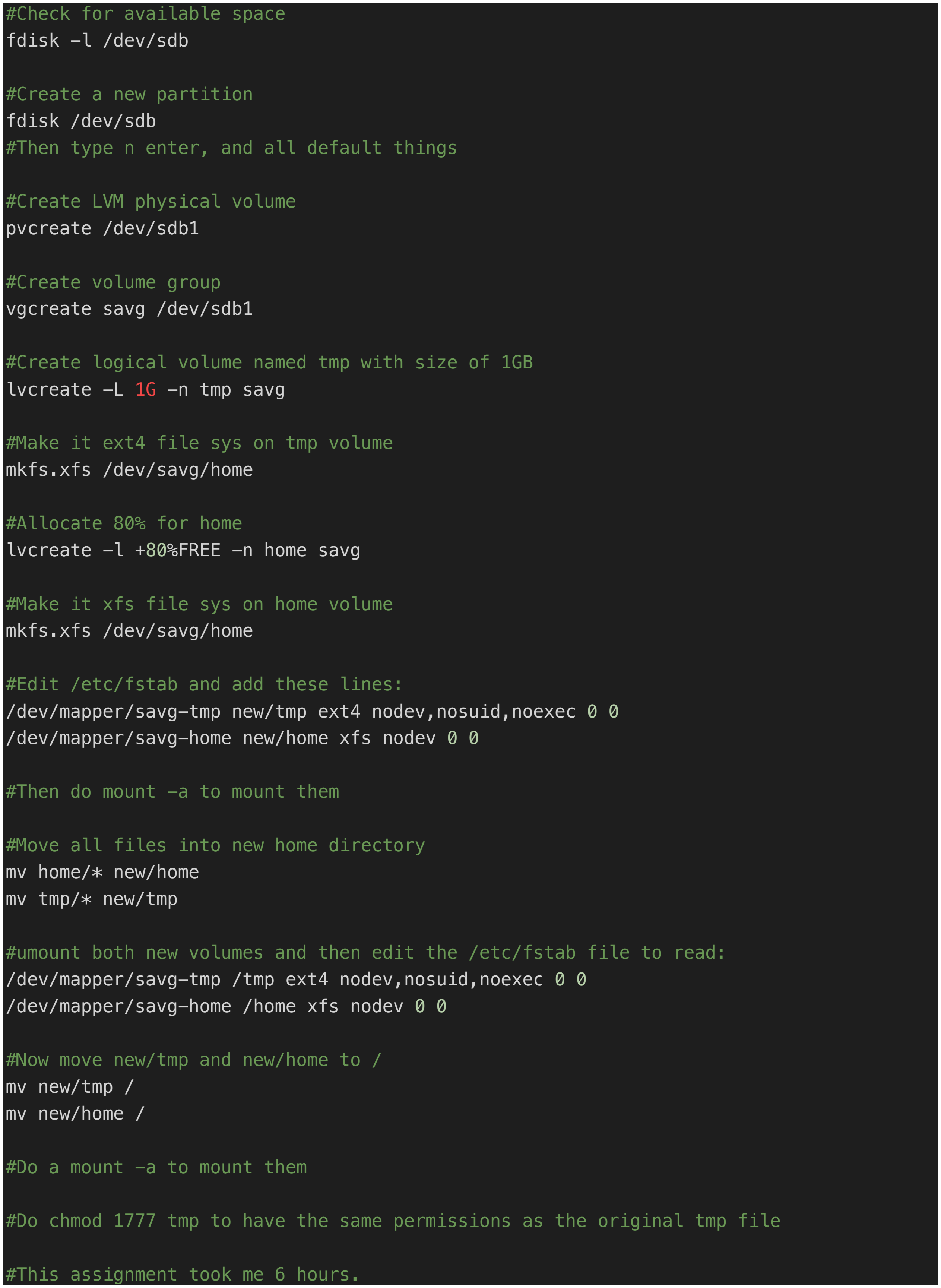
- Minlen = 10 → Min of 10 characters
- Dcredit = -2 → Must be at least 2 digits
- Ucredit = -2 → Must be at least 2 uppercase letters
- Ocredit = -1 → Must be at least 1 non-alphanumeric character
- Lcredit = 0 → Lowercase letters do not carry any credit

This is how I set up everything so specific users can login to specific machines and there is a password policy on every machine.

Time spent on assignment:
6 hours.

How I did it:
First, I installed Ansible.
And then in the ansible.cfg file, I added pipelining=true to basically enable anisble playbooks. I then configured the hosts file with saclass, containing all the machines with saclassDebian and saclassRocky also being added containing their respective machines. Then I created the dmuserplay playbook that set all the parameters for the groups and users. Basically how I did this is I used the already created modules: ansible.builtin.user and ansible.builtin.group. Thos attributes in those modules allowed me to check every user and group and set the attributes accordingly to make sure all the info was the same on all machines. Debian and Rocky hosts were separated but the only change in the code was checking for either wheel (Rocky) or sudo (Debian). It was just a couple of simple loops going through all the info and everything was added into the dmusers.yaml file.
For the umask.yaml playbook, I used the built in module ansible.builtin.copy. It was a very simple playbook; I simply gave the src and dest of the file which was the same because the file is in the same place on all machines. Group and user ownership was set to root with the owner and group flags. File permissions was set accordingly with the mode flag.

Time taken on assignment:
12 hours

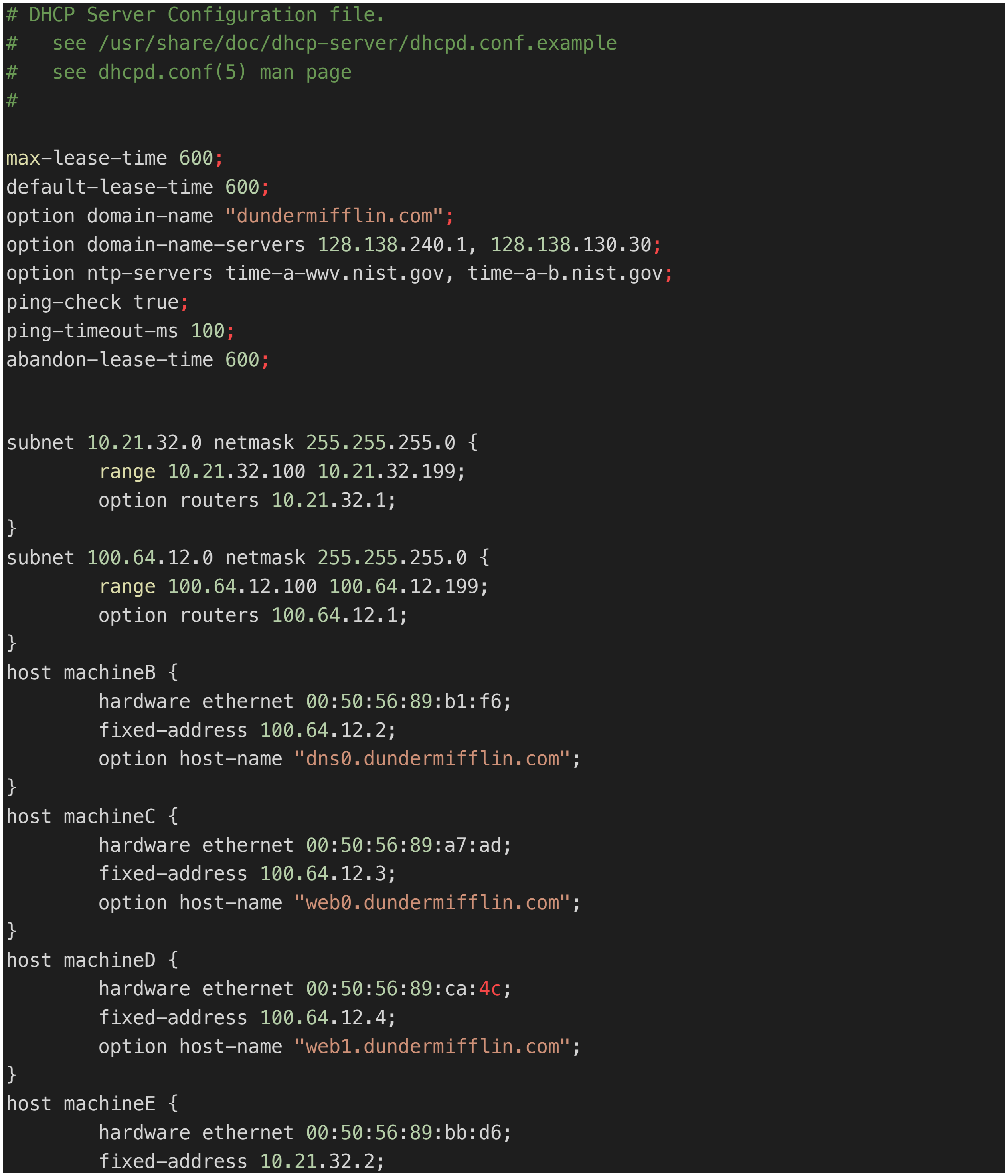


```
#Check for available space
fdisk -l /dev/sdb

#Create a new partition
fdisk /dev/sdb
#Then type n enter, and all default things

#Create LVM physical volume
pvcreate /dev/sdb1

#Create volume group
vgcreate savg /dev/sdb1

#Create logical volume named tmp with size of 1GB
lvcreate -L 1G -n tmp savg

#Make it ext4 file sys on tmp volume
mkfs.xfs /dev/savg/home

#Allocate 80% for home
lvcreate -l +80%FREE -n home savg

#Make it xfs file sys on home volume
mkfs.xfs /dev/savg/home

#Edit /etc/fstab and add these lines:
/dev/mapper/savg-tmp new/tmp ext4 nodev,nosuid,noexec 0 0
/dev/mapper/savg-home new/home xfs nodev 0 0

#Then do mount -a to mount them

#Move all files into new home directory
mv home/* new/home
mv tmp/* new/tmp

#umount both new volumes and then edit the /etc/fstab file to read:
/dev/mapper/savg-tmp /tmp ext4 nodev,nosuid,noexec 0 0
/dev/mapper/savg-home /home xfs nodev 0 0

#Now move new/tmp and new/home to /
mv new/tmp /
mv new/home /

#Do a mount -a to mount them

#Do chmod 1777 tmp to have the same permissions as the original tmp file

#This assignment took me 6 hours.
```

How I did it:

First, I installed the ISC-DHCP server onto Machine A.

Then I configured the DHCP server in the text file: /etc/dhcp/dhcpd.conf



```
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp-server/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#

max-lease-time 600;
default-lease-time 600;
option domain-name "dundermifflin.com";
option domain-name-servers 128.138.240.1, 128.138.130.30;
option ntp-servers time-a-wwv.nist.gov, time-a-b.nist.gov;
ping-check true;
ping-timeout-ms 100;
abandon-lease-time 600;


subnet 10.21.32.0 netmask 255.255.255.0 {
        range 10.21.32.100 10.21.32.199;
        option routers 10.21.32.1;
}
subnet 100.64.12.0 netmask 255.255.255.0 {
        range 100.64.12.100 100.64.12.199;
        option routers 100.64.12.1;
}
host machineB {
        hardware ethernet 00:50:56:89:b1:f6;
        fixed-address 100.64.12.2;
        option host-name "dns0.dundermifflin.com";
}
host machineC {
        hardware ethernet 00:50:56:89:a7:ad;
        fixed-address 100.64.12.3;
        option host-name "web0.dundermifflin.com";
}
host machineD {
        hardware ethernet 00:50:56:89:ca:4c;
        fixed-address 100.64.12.4;
        option host-name "web1.dundermifflin.com";
}
host machineE {
        hardware ethernet 00:50:56:89:bb:d6;
        fixed-address 10.21.32.2;
        option host-name "nfs.dundermifflin.com";
}
host machineF {
        hardware ethernet 00:50:56:89:1a:33;
        fixed-address 100.64.12.6;
        option host-name "dns1.dundermifflin.com";
}
```

The configuration is above.

To go over what it means, there is documentation that I followed that goes over basic commands that will set things like NTP servers, DNS, and ping timeout for machines that already have an IP. At the top the first few commands are global, meaning that all devices apply these commands. Things set were lease times, NTP servers, DNS servers, and the ping check. My settings show that ping check is enabled with a response wait time of 100 ms and a time out of 600 seconds or 10 minutes.

In the next section with the subnets, these are applied to the network specified. What I did was create a pool of Ips that will be given out to any other devices other than the current DM network.

And then, I configured host specific things like the hostname. A MAC-address and IP was needed to locate what machine it was so the DHCP server could set the hostname.

To configure Machine A manually:

- Typed: hostname router.dundermifflin.com to change the hostname
- Went into /etc/resolv.conf and added the domain and DNS Ips to the file
- Went into /etc/chrony.conf to add the NTP servers for ex: server time-awwv.nist.gov
    - I then commented out the public pool of NTP servers so the specified NTP servers were only set

To configure **Debian** based machines (web0 & dns1):

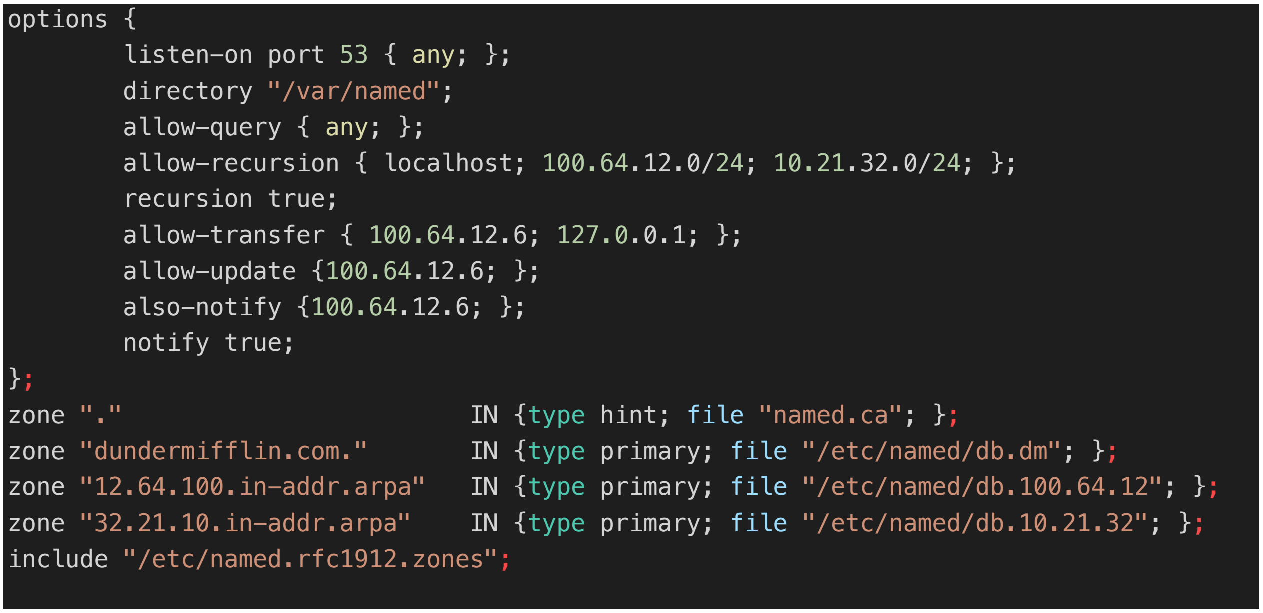
- Went into /etc/network/interfaces and added iface ens192 inet dhcp
- I installed ntpstats and ntp to declare ntp servers
- I went into /etc/ntpsec/ntp.conf and commented out all of the public pool ntp servers

To configure **RedHat** based machines (dns0, web1, & dns1):

- Went into nmtui and configured IPv4 config to automatic on that ethernet interface
- Removed the contents of /etc/hostname so that the DHCP server could update the hostnames accordingly
- Went into /etc/chrony.conf and commented out the public pool so that only the specified NTP servers from the DHCP server would be allowed

How I did it:
First I installed bind9 on dns1 and named on dns0. Both install the BIND dns server functionality.

DNS0:
First I went into /etc/namd.conf and configured the global options and zones:

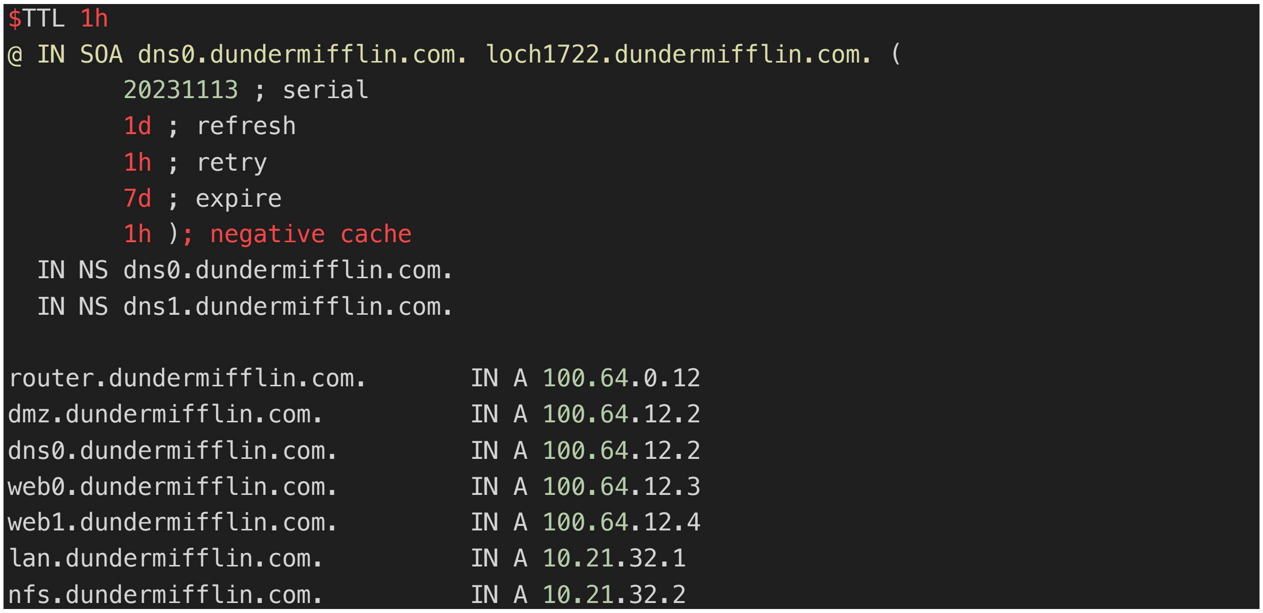


```
options {
        listen-on port 53 { any; };
        directory "/var/named";
        allow-query { any; };
        allow-recursion { localhost; 100.64.12.0/24; 10.21.32.0/24; };
        recursion true;
        allow-transfer { 100.64.12.6; 127.0.0.1; };
        allow-update {100.64.12.6; };
        also-notify {100.64.12.6; };
        notify true;
};
zone "."                        IN {type hint; file "named.ca"; };
zone "dundermifflin.com."       IN {type primary; file "/etc/named/db.dm"; };
zone "12.64.100.in-addr.arpa"   IN {type primary; file "/etc/named/db.100.64.12"; };
zone "32.21.10.in-addr.arpa"    IN {type primary; file "/etc/named/db.10.21.32"; };
include "/etc/named.rfc1912.zones";
```

At the bottom you can see the zones I specified, one forward DNS zone and two reverse DNS zones. To allow for dns1 to update their zones I did an allow-update on the dns1 IP.

I then went into /etc/named/ and created the 3 zone files: db.dm, db.100.64.12 and db.10.21.32

Db.dm:



```
$TTL 1h
@ IN SOA dns0.dundermifflin.com. loch1722.dundermifflin.com. (
        20231113 ; serial
        1d ; refresh
        1h ; retry
        7d ; expire
        1h ); negative cache
  IN NS dns0.dundermifflin.com.
  IN NS dns1.dundermifflin.com.

router.dundermifflin.com.       IN A 100.64.0.12
dmz.dundermifflin.com.          IN A 100.64.12.2
dns0.dundermifflin.com.         IN A 100.64.12.2
web0.dundermifflin.com.         IN A 100.64.12.3
web1.dundermifflin.com.         IN A 100.64.12.4
lan.dundermifflin.com.          IN A 10.21.32.1
nfs.dundermifflin.com.          IN A 10.21.32.2
```

```
dns1.dundermifflin.com.         IN A 100.64.12.6
bsd.dundermifflin.com.          IN A 100.64.12.7
machinea.dundermifflin.com. 7d  IN CNAME router.dundermifflin.com.
machineb.dundermifflin.com. 7d  IN CNAME dns0.dundermifflin.com.
machinec.dundermifflin.com. 7d  IN CNAME web0.dundermifflin.com.
machined.dundermifflin.com. 7d  IN CNAME web1.dundermifflin.com.
machinee.dundermifflin.com. 7d  IN CNAME nfs.dundermifflin.com.
machinef.dundermifflin.com. 7d  IN CNAME dns1.dundermifflin.com.
machinex.dundermifflin.com. 7d  IN CNAME bsd.dundermifflin.com.
;dundermifflin.com.                 5m  IN CNAME web0.dundermifflin.com. ; CHANGED
THIS ONE
dundermifflin.com.          5m  IN A 100.64.12.3
www.dundermifflin.com.      5m  IN CNAME web0.dundermifflin.com.
www1.dundermifflin.com.     5m  IN CNAME web1.dundermifflin.com.
dns.dundermifflin.com.      5m  IN CNAME dns0.dundermifflin.com.
files.dundermifflin.com.    7d  IN CNAME nfs.dundermifflin.com.
```

db.100.64.12:

```
$TTL 1h
@ IN SOA dns0.dundermifflin.com. loch1722.dundermifflin.com. (
        20231113 ; serial
        1d ; refresh
        1h ; retry
        7d ; expire
        1h ); negative cache
  IN NS dns0.dundermifflin.com.
  IN NS dns1.dundermifflin.com.

1 IN PTR dmz.dundermifflin.com.
2 IN PTR dns0.dundermifflin.com.
3 IN PTR web0.dundermifflin.com.
4 IN PTR web1.dundermifflin.com.
6 IN PTR dns1.dundermifflin.com.
7 IN PTR bsd.dundermifflin.com
```

Db.10.21.32:

```
$TTL 1h
@ IN SOA dns0.dundermifflin.com. loch1722.dundermifflin.com. (
        20231113 ; serial
        1d ; refresh
        1h ; retry
        7d ; expire
        1h ); negative cache
  IN NS dns0.dundermifflin.com.
  IN NS dns1.dundermifflin.com.

1 IN PTR lan.dundermifflin.com.
2 IN PTR nfs.dundermifflin.com.
```

For all of these files, the configuration is pretty self-explanatory, with specifications of SOA and declaring the 2 NS DNS servers. For forward DNS, I found that the CNAME from dundermifflin.com to web0.dundermifflin.com was not valid so I switched it to an A record and replaced web0.* with the IP of web0. Other than that, all A and CNAME records are listed, and reverse DNS records are listed as well respectively above.

DNS1:
First, I went into /etc/bind/named.conf and created the conf file:

```
options {
        listen-on port 53 { any; };
        directory "/var/cache/bind";
        allow-query { any; };
        allow-recursion { localhost; 100.64.12.0/24; 10.21.32.0/24; };
        recursion true;
    allow-transfer { none};
    //allow-update { 100.64.12.2; };
    notify false;
};
include "/etc/bind/named.conf.default-zones";
zone "dundermifflin.com."   IN {type secondary;primaries {100.64.12.2;};file
"/etc/named/db.dm";};
zone "12.64.100.in-addr.arpa"   IN {type secondary;primaries {100.64.12.2;};file
"/etc/named/db.100.64.12";};
zone "32.21.10.in-addr.arpa"    IN {type secondary;primaries {100.64.12.2;};file
"/etc/named/db.10.21.32";};
```

It is very similar to DNS0s config, but I specify the type as secondary and point to the IP where the primary is.

Finally on Machine A, I went into /etc/resolve.conf and changes the name servers to these two DNS servers. I then went into /etc/dhcpd/dhcpd.conf and changed the nameservers to these two DNS servers. I rebooted all machines and was finished.

Time spent on assignment:
8 hours

How I did Lab 11:

- Intalled chrony on RedHat machines:
  - Dnf install chronyc
- On Machine A:
  - Int /etc/chrony.conf, I edited to only have both NIST time servers and only allow for LAN and DMZ subnets to access these servers:

```
server time-a-wwv.nist.gov iburst
server time-a-b.nist.gov iburst

allow 10.21.32.0/24
allow 100.64.12.0/24
```

  - In /etc/dhcp/dhcpd.conf I configured the ntp servers now according to the subnet IPs instead of one global rule for both NIST servers. So the option ntp-servers was set in the subnet for 100.64.12.0/24 and 10.21.32.0/24.
- On Machine E:
  - To install NFS, I did: dnf install nfs-utils
  - In the /etc/exports file I configured:

```
       /home/accounting/www 100.64.12.0/24(rw,sync,root_squash,no_all_squash)
```

  - I created the directory: /home/accounting/www
- On Machine C:
  - To install NFS, I did: apt install nfs-common
  - I mounted the NFS server by doing:

```
       mount -t nfs -o ro,soft
10.21.32.2:/home/accounting/www/var/www/html/dundermifflin/accounting/
```

  - To make this mount permanent I went into /etc/fstab and entered this:

```
   10.21.32.2:/home/accounting/www /var/www/html/dundermifflin/accounting nfs
ro,soft 0 0
```

  - Also, in /etc/group, I added www-data to the accounting group so it could access the web files in accounting and display it on the website.
- On Machine D:
  - To install NFS, I did: dnf install nfs-utils
  - Everything above was copied for this machine. Instead of adding www-data, it is called 'apache' which was added to the accounting group
  - In crontab -e, I changed the rsync command to get the dundermifflin content from machine c without copying over accounting. This is the new input:

```
*/5 * * * * rsync -aH --exclude='accounting/'
100.64.12.3:/var/www/html/dundermifflin/* /var/www/html/dundermifflin
```

- Once all of this was done, I did a reboot of all devices, starting with A and then the rest.

Time taken on Lab: 7 hours.

How I did the Lab:

First I went onto all machines and did:
Systemctl start nftables
Systemctl enable nftables

I then added the nftables.conf file provided for us on canvas to machine A in /etc/sysconfig/

I then created and added these nftables.conf files on the associated machines:

Machine A:

```
#!/usr/sbin/nft -f

flush ruleset

#  Set your DMZ net here
define DMZ = 100.64.12.0/24

#  Machine A
table ip saclass {
   #  Incoming chain
   chain incoming {
      #  Default drop
      type filter hook input priority 0; policy drop;
      #  accept loopback
      iifname lo accept
      #  established connections
      ct state invalid drop
      ct state related,established accept
      #  saclass grader and proxy
      tcp dport {4113,4114} accept
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  ssh from LAN, WAN, DMZ and VPN
      ip saddr {10.21.32.0/24,100.64.0.0/24,$DMZ,198.11.0.0/16} tcp dport 22 accept
      #  Incoming DHCP and NTP
      udp dport {67,123} accept
   }
   #  Outgoing chain
   chain outgoing {
      #  Default accept
      type filter hook output priority 0; policy accept;
      #  Block facebook
      ip daddr 157.240.28.35 drop
   }
   #  Forward chain
   chain forwarding {
      # Default drop
      type filter hook forward priority 0; policy drop;
      #  established connections
      ct state invalid drop
      ct state related,established accept
      #  interface based chains
      iifname "ens192" oifname "ens224" jump WAN2DMZ
      iifname "ens192" oifname "ens256" jump WAN2LAN
      iifname "ens224" oifname "ens192" jump DMZ2WAN
      iifname "ens224" oifname "ens256" jump DMZ2LAN
      iifname "ens256" oifname "ens192" jump LAN2WAN
      iifname "ens256" oifname "ens224" jump LAN2DMZ
   }
   #  WAN to DMZ chain
   chain WAN2DMZ {
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  DNS
      udp dport 53 accept;
      #  ssh, html, grader
      tcp dport {22,80,4113} accept;
   }
   #  WAN to LAN chain
   chain WAN2LAN {
      # only return traffic
   }
   #  DMZ to WAN
   chain DMZ2WAN {
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  DNS
      udp dport 53 accept;
      #  DNS, http, https
      tcp dport {53,80,443} accept;
   }
   #  DMZ to LAN
   chain DMZ2LAN {
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  ssh and NFS
      tcp dport {22,2049} accept;
   }
   #  LAN to DMZ
   chain LAN2DMZ {
      #  Allow everything
      ip saddr {10.21.32.0/24} accept;
   }
   #  LAN to WAN
   chain LAN2WAN {
      #  Block facebook
      ip daddr 157.240.28.35 drop
      #  Allow everything else
      ip saddr {10.21.32.0/24} accept;
   }
}
#  NAT LAN to WAN
table ip nat {
   chain POSTROUTING {
      type nat hook postrouting priority srcnat; policy accept;
      oifname "ens192" ip saddr 10.21.32.0/24 masquerade
   }
}
```

Machines B & F:

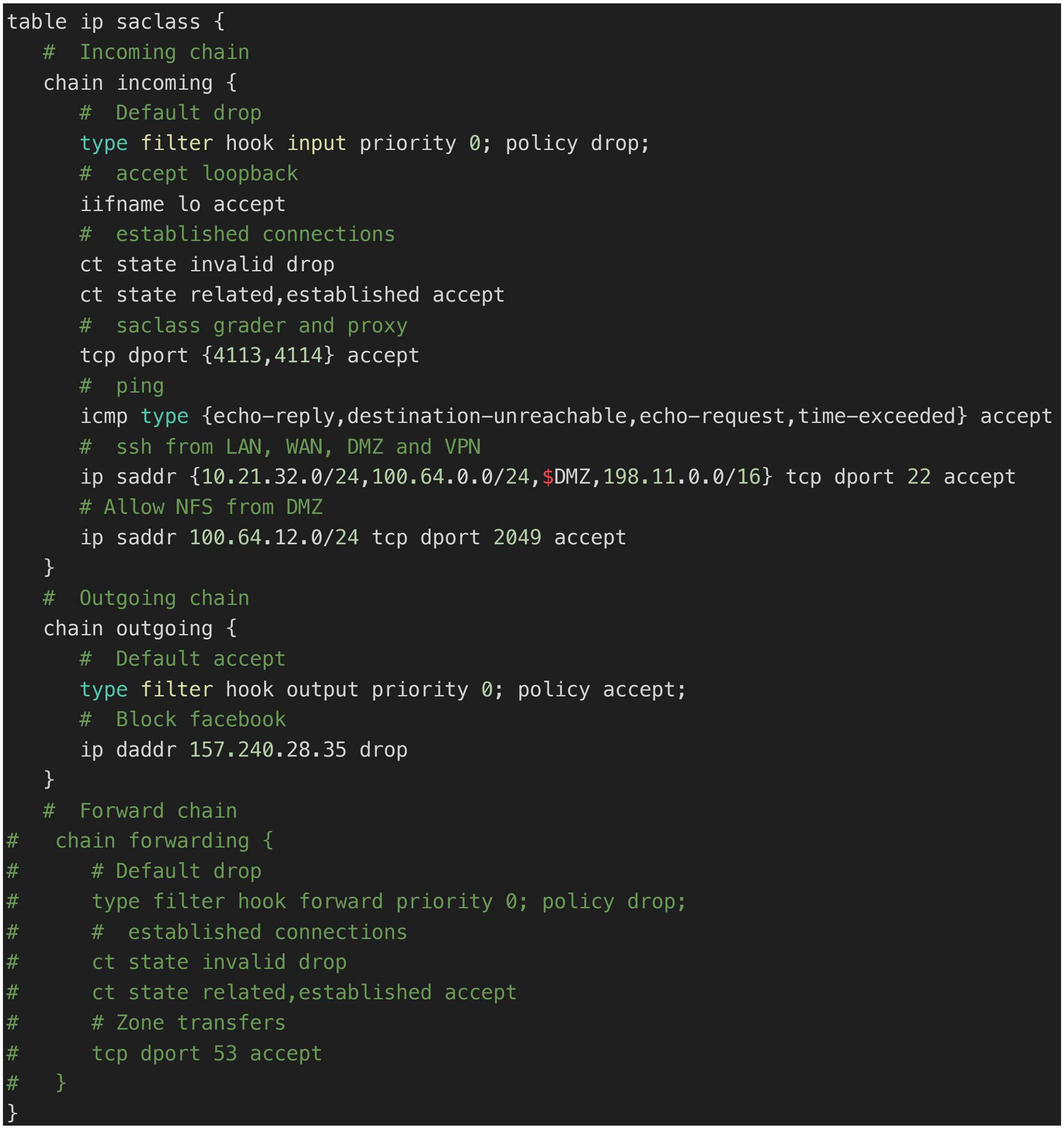
```
#!/usr/sbin/nft -f

flush ruleset

#  Set your DMZ net here
define DMZ = 100.64.12.0/24

#  Machine A
table ip saclass {
   #  Incoming chain
   chain incoming {
      #  Default drop
      type filter hook input priority 0; policy drop;
      #  accept loopback
      iifname lo accept
      #  established connections
      ct state invalid drop
      ct state related,established accept
      #  saclass grader and proxy
      tcp dport {4113,4114} accept
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  ssh from LAN, WAN, DMZ and VPN
      ip saddr {10.21.32.0/24,100.64.0.0/24,$DMZ,198.11.0.0/16} tcp dport 22 accept
      #  Incoming DNS
      udp dport 53 accept
      tcp dport 53 accept
   }
   #  Outgoing chain
   chain outgoing {
      #  Default accept
      type filter hook output priority 0; policy accept;
      #  Block facebook
      ip daddr 157.240.28.35 drop
   }
   #  Forward chain
#   chain forwarding {
#      # Default drop
#      type filter hook forward priority 0; policy drop;
#      #  established connections
#      ct state invalid drop
#      ct state related,established accept
#      # Zone transfers
#      tcp dport 53 accept
#   }
}
```

Machines C & D:

```
#!/usr/sbin/nft -f

flush ruleset

#  Set your DMZ net here
define DMZ = 100.64.12.0/24

#  Machine A
table ip saclass {
   #  Incoming chain
   chain incoming {
      #  Default drop
      type filter hook input priority 0; policy drop;
      #  accept loopback
      iifname lo accept
      #  established connections
      ct state invalid drop
      ct state related,established accept
      #  saclass grader and proxy
      tcp dport {4113,4114} accept
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  ssh from LAN, WAN, DMZ and VPN
      ip saddr {10.21.32.0/24,100.64.0.0/24,$DMZ,198.11.0.0/16} tcp dport 22 accept
      # Allow incoming HTTP & HTTPS
      tcp dport {80, 443} accept
   }
   #  Outgoing chain
   chain outgoing {
      #  Default drop
      type filter hook output priority 0; policy drop;
      # accept loopback
      oifname lo accept
      # established connections
      ct state invalid drop
      ct state related,established accept
      #  Block facebook
      ip daddr 157.240.28.35 drop
      # Allow DHCP
      ip daddr 100.64.12.1 udp dport 67 accept
      # Allow NTP
      ip daddr 100.64.12.1 udp dport 123 accept
      # Allow DNS to B and F
      ip daddr 100.64.12.2 udp dport 53 accept
      ip daddr 100.64.12.6 udp dport 53 accept
      # Allow NFS to E
      ip daddr 10.21.32.2 tcp dport 2049 accept
      # Allow SSH to DMZ
      ip daddr 100.64.12.0/24 tcp dport 22 accept
      # Allow ping except LAN
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} ip
daddr !=10.21.32.0/24 accept
      # Allow http/https to anywhere
      tcp dport {80, 443} accept
   }
   #  Forward chain
#   chain forwarding {
#      # Default drop
#      type filter hook forward priority 0; policy drop;
#      #  established connections
#      ct state invalid drop
#      ct state related,established accept
#      # Zone transfers
#      tcp dport 53 accept
#   }
}
```

Machine E:

```
#!/usr/sbin/nft -f

flush ruleset

#  Set your DMZ net here
define DMZ = 100.64.12.0/24

#  Machine A
```



```
table ip saclass {
   #  Incoming chain
   chain incoming {
      #  Default drop
      type filter hook input priority 0; policy drop;
      #  accept loopback
      iifname lo accept
      #  established connections
      ct state invalid drop
      ct state related,established accept
      #  saclass grader and proxy
      tcp dport {4113,4114} accept
      #  ping
      icmp type {echo-reply,destination-unreachable,echo-request,time-exceeded} accept
      #  ssh from LAN, WAN, DMZ and VPN
      ip saddr {10.21.32.0/24,100.64.0.0/24,$DMZ,198.11.0.0/16} tcp dport 22 accept
      # Allow NFS from DMZ
      ip saddr 100.64.12.0/24 tcp dport 2049 accept
   }
   #  Outgoing chain
   chain outgoing {
      #  Default accept
      type filter hook output priority 0; policy accept;
      #  Block facebook
      ip daddr 157.240.28.35 drop
   }
   #  Forward chain
#   chain forwarding {
#      # Default drop
#      type filter hook forward priority 0; policy drop;
#      #  established connections
#      ct state invalid drop
#      ct state related,established accept
#      # Zone transfers
#      tcp dport 53 accept
#   }
}
```