NASA HW6

b09902004 郭懷元

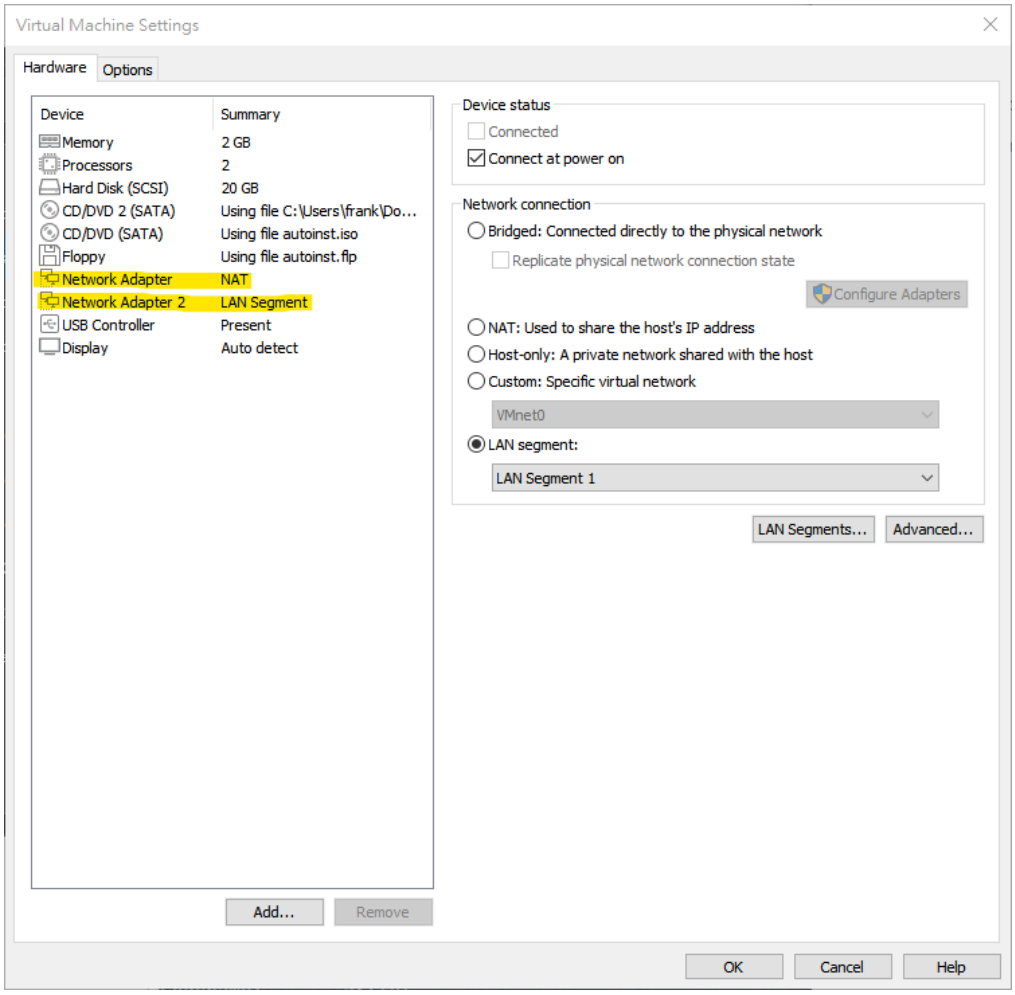
Network Administration

1. DNS & DHCP

> Refs:
>
> DNS lab slides
> https://magiclen.org/ubuntu-start-job-wait-network/
> https://ubuntu.com/server/docs/network-dhcp
> https://ithelp.ithome.com.tw/articles/10030241

Server VM setup

Hypervisor: VMWare
In the hardware setting, give the VM a NAT adapter and a LAN segment adapter.



The NAT adapter is the interface to outer network, and the LAN segment adapter is for internal network.

OS installation & static IP configuration

OS: Ubuntu Server 20.04

During the installation steps, just follow the default options.
After the installation and reboot, check which interface is the internal one:

```
ip a
```

The one without `inet` is the LAN segment interface, in my case it's `ens34`.

Check which file keeps the config for `ens34`:

```
cd /etc/netplan; grep -r ens34
```

Turns out it's in `00-installer-config.yaml`

Edit only the `ens34` part of `00-installer-config.yaml` like this:

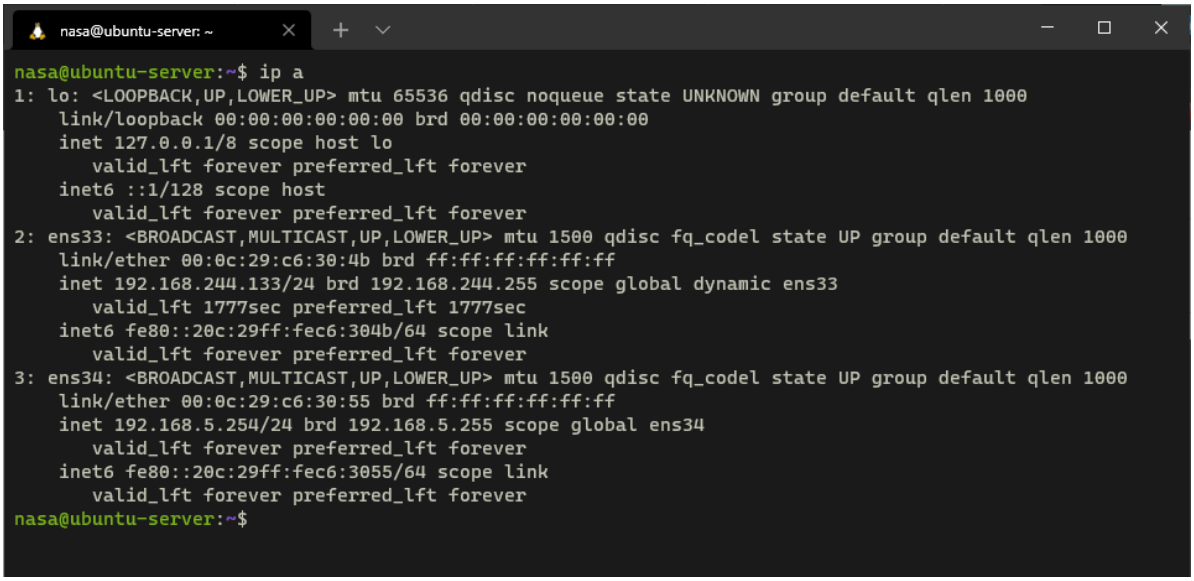
```
ens34:
      addresses: ["192.168.5.254/24"]
```

Then regenerate the network config file for systemd and apply it:

```
sudo netplan generate && sudo netplan apply
```

Check interfaces:

```
ip a
```



DHCP configuration

Install `dhcpd`

```
sudo apt install isc-dhcp-server
```

Edit `/etc/dhcp/dhcpd.conf`

```
default-lease-time 600;
max-lease-time 7200;
ddns-update-style none;
authoritative;
subnet 192.168.5.0 netmask 255.255.255.0 {
        range 192.168.5.100 192.168.5.200;
        option routers 192.168.5.254;
        option domain-name-servers 192.168.5.254;
}
```

`subnet 192.168.5.0 netmask 255.255.255.0` specifies the subnet.
`range` is the range of IP addresses that will be given to the client.
`option routers` is the default router/gateway our clients will have.
`option domain-name-servers` is the DNS server our clients will have.

Edit `/etc/default/isc-dhcp-server`

```
INTERFACESv4="ens34"
INTERFACESv6=""
```

`ens34` is the interface to the internal network.

Finally, restart `dhcpd`

```
sudo systemctl restart isc-dhcp-server.service
```

DNS configuration

Install `bind9`, then `cd` to the folder with configs.

```
sudo apt install bind9
cd /etc/bind
sudo mkdir zones
```

Edit `named.conf.options`:

```
options {
        directory "/var/cache/bind";
        dnssec-validation auto;
        allow-query { any; };
        listen-on { 192.168.5.254; };
        allow-recursion { any; };
};
```

The first two lines are just defaults.
`allow-query` sets who's DNS queries are allowed.
`listen-on` sets on which interface (specified by IP address) the DNS service will listen for queries.
`forwarders` are the DNS servers we will forward non-local.
`allow-recursion` sets who's DNS queries are allowed to do recursive query.

Edit `named.conf.options`:

```
zone "b09902004.com" {
        type master;
        file "/etc/bind/zones/named.hosts";
};

zone "3.2.1.in-addr.arpa" {
        type master;
        file "/etc/bind/zones/named.rev";
};
```

`zone "b09902004.com"` is for using domain name to look up IP address.
`zone "3.2.1.in-addr.arpa"` is for using IP address to look up domain name.
Note that `3.2.1` is the reverse of first 3 numbers of the address `1.2.3.4`.

Create `zones/named.hosts`:

```
$TTL    604800
@       IN      SOA     ns.b09902004.com.       root.b09902004.com. (
        3               ; Serial
        604800          ; Refresh
        86400           ; Retry
        2419200         ; Expire
        604800 )        ; Negative Cache TTL
                        IN      NS      ns.b09902004.com.
ns                      IN      A       192.168.5.254
www                     IN      A       1.2.3.4
```

Create `zones/named.rev`:

```
$TTL    604800
@       IN      SOA     ns.b09902004.com.       root.b09902004.com. (
                        3               ; Serial
                        604800          ; Refresh
                        86400           ; Retry
                        2419200         ; Expire
                        604800 )        ; Negative Cache TTL
        IN      NS      ns.b09902004.com.
4       IN      PTR     www.b09902004.com.
```

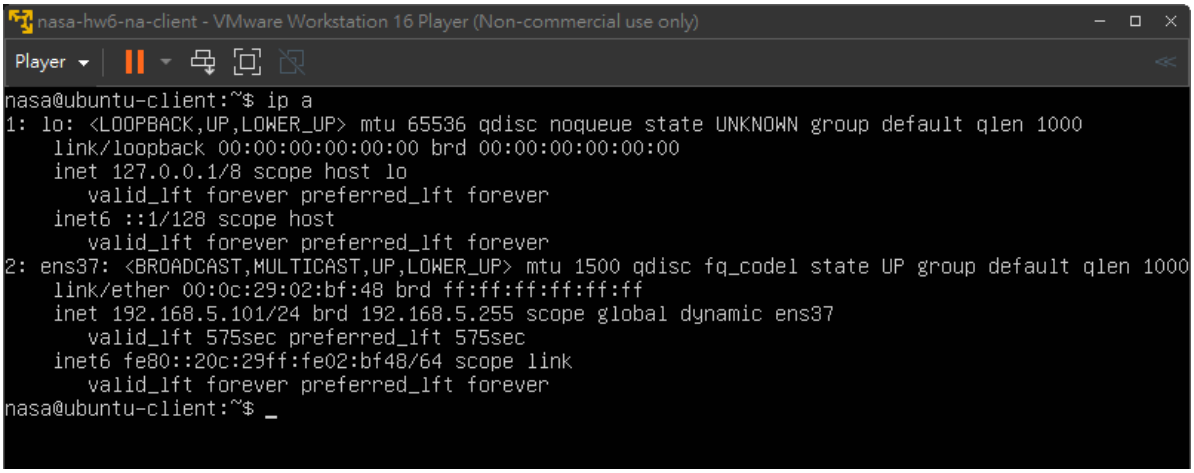
Finally, check the configuration and restart `bind`

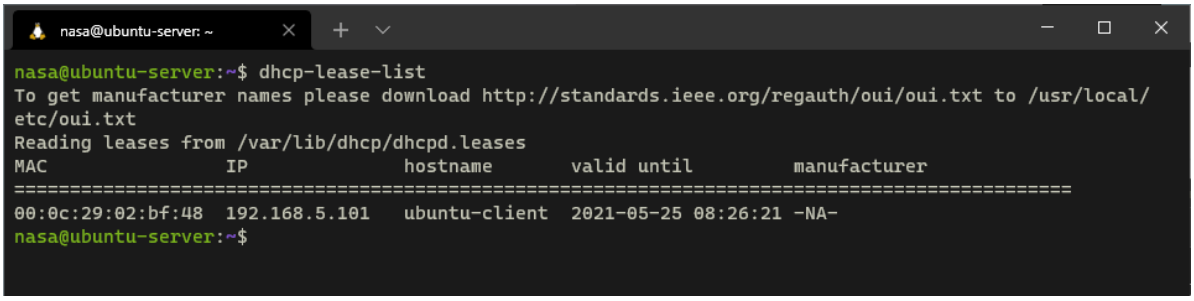
```
sudo named-checkconf
sudo service bind9 reload
```

Client VM setup

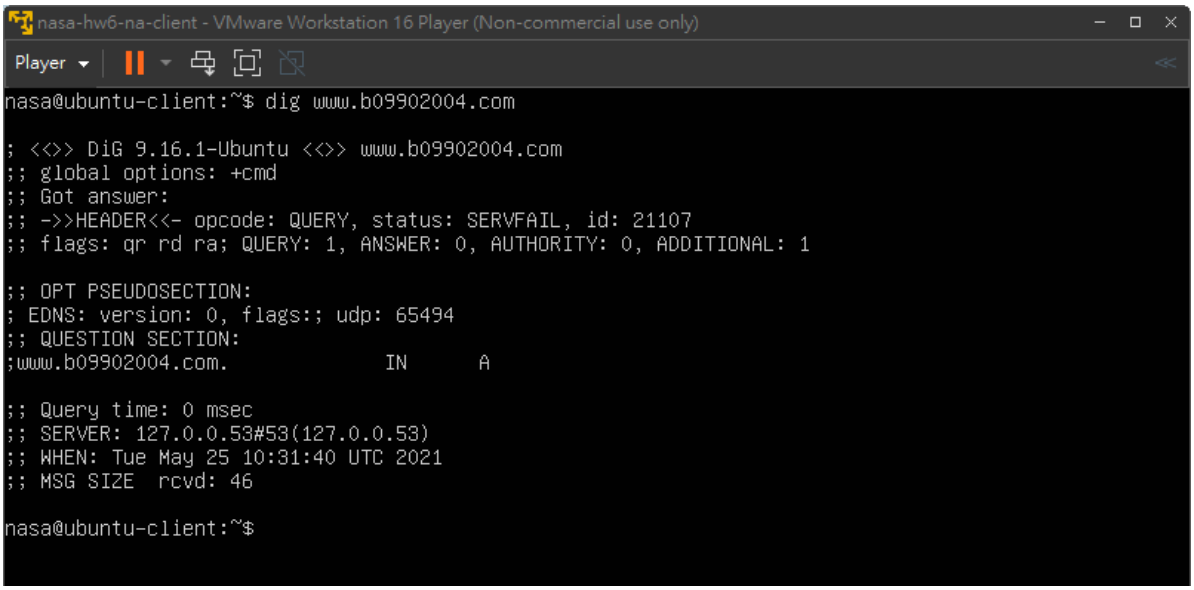
Basically the same as the server. Still add two network interfaces because we need the NAT interface for the installation process.
After the OS is installed, the NAT interface can be removed so that test result isn't influenced.

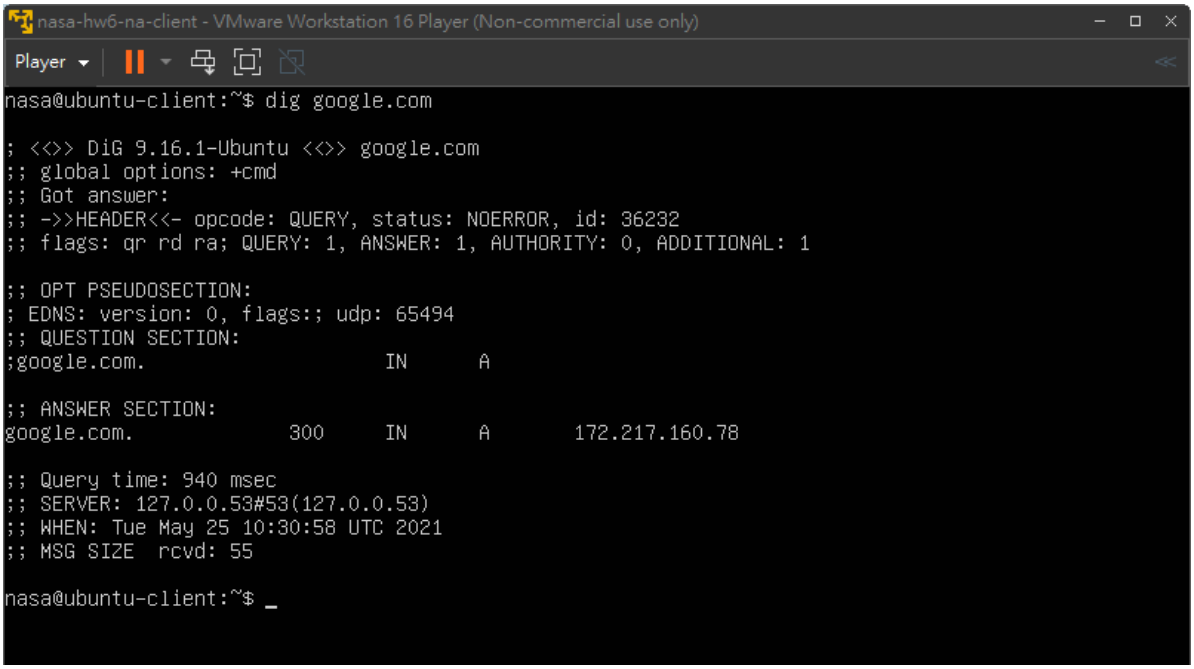
DHCP result

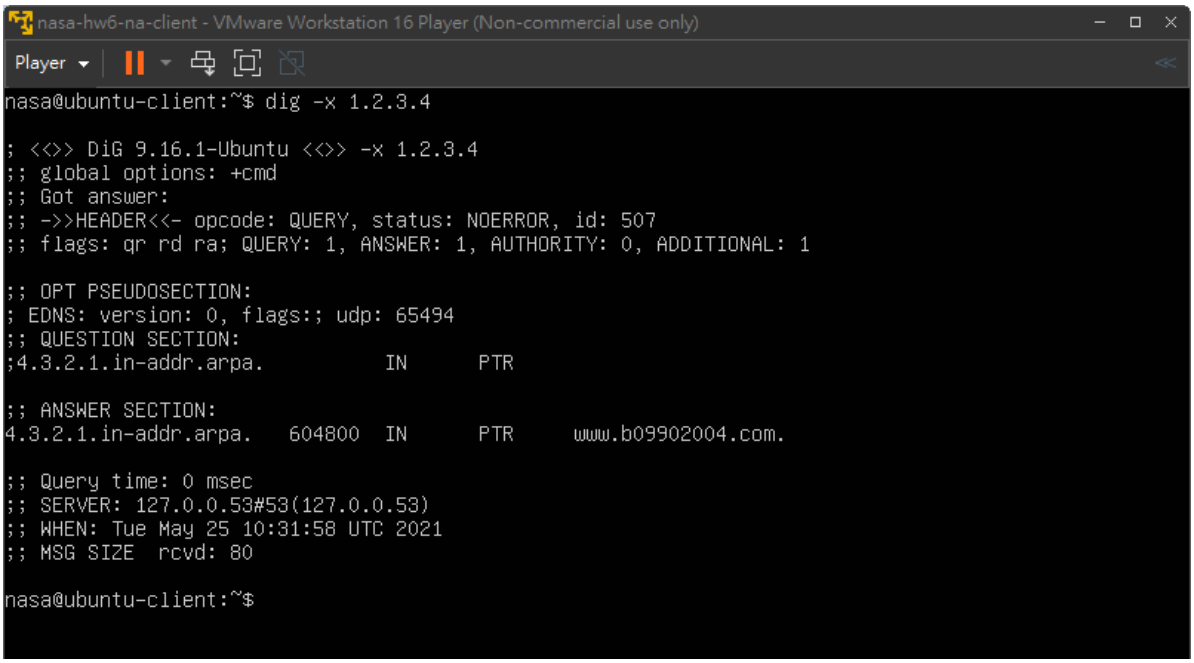

nasa-hw6-na-client - VMware Workstation 16 Player (Non-commercial use only)
Player
nasa@ubuntu-client:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
link/ether 00:0c:29:02:bf:48 brd ff:ff:ff:ff:ff:ff
inet 192.168.5.101/24 brd 192.168.5.255 scope global dynamic ens37
valid_lft 575sec preferred_lft 575sec
inet6 fe80::20c:29ff:fe02:bf48/64 scope link
valid_lft forever preferred_lft forever
nasa@ubuntu-client:~$ _


nasa@ubuntu-server: ~
nasa@ubuntu-server:~$ dhcp-lease-list
To get manufacturer names please download http://standards.ieee.org/regauth/oui/oui.txt to /usr/local/etc/oui.txt
Reading leases from /var/lib/dhcp/dhcpd.leases
MAC IP hostname valid until manufacturer
00:0c:29:02:bf:48 192.168.5.101 ubuntu-client 2021-05-25 08:26:21 -NA-
nasa@ubuntu-server:~$

DNS result


nasa-hw6-na-client - VMware Workstation 16 Player (Non-commercial use only)
Player
nasa@ubuntu-client:~$ dig www.b09902004.com
; <<>> DiG 9.16.1-Ubuntu <<>> www.b09902004.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 21107
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.b09902004.com. IN A
;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue May 25 10:31:40 UTC 2021
;; MSG SIZE rcvd: 46
nasa@ubuntu-client:~$





---

2. Short Answer

1.

> Refs:
>
> None

The TTL of a DNS record is the time a cache of that record will live.
A longer TTL usually means a longer DNS propagation time.
A long TTL can reduce the load of name servers. A short TTL can keep more clients with the latest DNS record.

2.

> Refs:
>
> https://en.wikipedia.org/wiki/Domain_Name_System

Without cache, every time a DNS server receives a query from client, it will recursively ask the right name server until it gets the answer from the authoritative server.
If the DNS server is using cache, when a client queries a domain that is in the cache, it will return the cached record.

Using cache reduces the workload of authoritative servers, since DNS resolvers query less times.

3.

> Refs:
>
> http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

A DNS cache poisoning attack injects fake DNS records to the victim DNS server's cache. A fake record might point commonly-used domains to the attacker's machine, allowing other attacks such as man-in-the-middle attack to happen.

A DNS cache poisoning attack can be carried out this way:

1. The attacker sends a DNS query for a hostname he wants to hijack (say `example.net`) to the victim name server.
2. The victim server doesn't have a cache for that domain, so it starts a recursive query for `example.net`.
3. Meanwhile, the attacker floods the victim with DNS responses of `example.net`.
4. If one of the responses matches the query ID, goes to the right port, and arrives before the genuine server, the attack is done.

A mitigation is to use random query IDs instead of incremental IDs (used in the old days).

4.

> Refs:
>
> http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

The original cache poisoning attack requires the hosname the attacker wants to hijack not to be in the cache. However in Kaminsky attack, the attacker sends a query for a new, randomized hostname within the target domain. Therefore the attacker can keep guessing query ID without obsticles.

One defense is to randomize the source port, making the attack harder but still possible.
Another defense is to use certificates to verify that responses are coming from genuine servers.

System Administration

1. This Problem Is Not For Sale

1.1

> Refs:
>
> https://qizhanming.com/blog/2018/08/08/how-to-install-nfs-on-centos-7

flag: `NASA{M0un71n6_NF5!2021}`

On workstation

Create a `.qcow2` disk for VM

```
mkdir -p /tmp2/b09902004/img
cd /tmp2/b09902004/img
qemu-img create -f qcow2 nfs.qcow2 10G
```

Install VM with `virt-install` (basically copy-paste from previous homework)

```
virt-install \
--name centos-1 \
--ram 2048 \
--vcpus 2 \
--disk /tmp2/b09902004/img/nfs.qcow2 \
--location http://centos.cs.nctu.edu.tw/7.9.2009/os/x86_64/ \
--extra-args="console=tty0 console=ttyS0,115200n8" \
--nographics
```

On VM

Install `nfs-utils`, setup the firewall

```
yum install -y nfs-utils
systemctl enable --now rpcbind
firewall-cmd --permanent --add-service={rpc-bind,mountd,nfs}
firewall-cmd --reload
```

Mount and get the flag

```
mount.nfs 10.217.44.112:/e/NASA_flag /mnt
cd /mnt
cat flag
```

1.2

> Refs:
>
> `man exports`
> http://linux.vbird.org/linux_enterprise/kerberos.php

In `/etc/exports`, there is an option `sec=` that can use kerberos to authenticate hosts and encrypt nfs traffic (nfs data is transferred in plain text). A few options such as `ro`, `rw`, `root_squash` can be set differently based on `sec` level.

The `/etc/exports` at the server probably makes all hosts without kerberos permission unable to access all directories under `/e/` except `/e/NASA_flag/`, therefore we can't mount `/e/undergrads` to gain access to other people's home directory.

1.3

> Refs:
>
> https://www.osc.edu/book/export/html/4523
> https://www.alibabacloud.com/help/zh/doc-detail/143009.htm

File path: `/home/student/09/b09902004/.nasa-is-an-awesome-course`

Create `.nasa-is-an-awesome-course`:

```
touch .nasa-is-an-awesome-course
```

Make the file only accessible to myself first:

```
chmod 600 .nasa-is-an-awesome-course
```

Get the uid of `wp` and use `NFSv4 ACL` to control more advanced file permission:

```
id wp
nfs4_setfacl -a A::69465:RW .nasa-is-an-awesome-course
```

1.4

> Refs:
>
> https://en.wikipedia.org/wiki/Network_File_System
> https://zh.wikipedia.org/wiki/ISCSI

Comparison

NFS	iSCSI
Allow access to storage over network	Same
Server serves files to client	Server serves blocks to client
NFSv4 is stateful	Requires a higher level lock manager for concurrency
No choice for filesystem	Any filesystem you like with favored features

Use Case

NFS: File sharing between different machines, such as user home directories on workstation.

iSCSI: A single server accessing a room of storage devices.

2. Getting Your Fix of VMs

2.1

> Refs:
>
> https://www.linux.com/training-tutorials/how-rescue-non-booting-grub-2-linux/
> https://unix.stackexchange.com/questions/44027/how-to-fix-boot-failure-due-to-incorrect-fstab
> https://wiki.archlinux.org/title/Device_file#virtio-blk

- Missing `/boot/grub/grub.cfg`
- Incorrect `/etc/fstab` causing mounting problem

In GRUB CLI

First, list the partitions:

```
grub> ls
```

In the output we see `(hd0,msdos1)`, that's probably where the OS is installed

Check what's in that partition:

```
grub> ls (hd0,msdos1)/
grub> ls (hd0,msdos1)/boot
grub> ls (hd0,msdos1)/boot/grub
```

The output of second line shows us the kernel and image files we need later.
The output of third line shows that `grub.cfg` is missing, that's why grub didn't find the os.

Boot:

```
grub> set root=(hd0,1)
grub> linux /boot/vmlinuz-linux root=/dev/vda1
grub> initrd /boot/initramfs-linux.img
grub> boot
```

`root=/dev/vda1` is the location of the root filesystem. Since it's a `.qcow2` disk, it's probably `/dev/vd*`. And because it's the only device, the name should be `/dev/vda`. (Or you can use UUID to mount)

Our first boot try failed, and looking at the error message that popped up, it's because the OS failed to mount something to `/mnt`.

Add an option then boot:

```
grub> set root=(hd0,1)
grub> linux /boot/vmlinuz-linux root=/dev/vda1 init=/bin/bash
grub> initrd /boot/initramfs-linux.img
grub> boot
```

The `fstab` has some problematic lines causing mounting failure. In order to bypass that, `init=/bin/bash` is added.
We are now in linux!

In Linux

Remount root folder to gain write access:

```
mount -o remount,rw /
```

Reconfigure grub:

```
grub-install --target=i386-pc /dev/vda
grub-mkconfig -o /boot/grub/grub.cfg
```

Fix `/etc/fstab` by deleting or commenting this line:

```
/some-filesystem.img /mnt ext4 rw,noatime
```

Finally, reboot the system by pressing `CTRL` + `ALT` + `DEL`.

2.2

> Refs:
>
> `mdadm` man page
> https://www.reddit.com/r/linuxquestions/comments/debx7w/mdadm_raid0_default_layout/
> https://wiki.archlinux.org/title/Mkinitcpio#Image_creation_and_activation
> https://wiki.archlinux.org/title/Install_Arch_Linux_on_LVM#Adding_mkinitcpio_hooks
> https://wiki.archlinux.org/title/RAID#Update_configuration_file
> https://wiki.archlinux.org/title/RAID#Installing_Arch_Linux_on_RAID

2.2.1

Fix RAID array to `chroot`

The system seems to be messed up completely, so we have to boot with an iso first.
After it's booted, check the disk status first:

```
lsblk
```

`md0` is missing in the list, so check its raid status:

```
mdadm --detail /dev/md0
```

According to the output, both disks are working, but the array some how failed to start. Also, the two devices in the array are `/dev/dm-0` and `/dev/vda5`.

Manually assemble the array and mount them:

```
echo 2 > /sys/module/raid0/parameters/default_layout
mdadm --stop /dev/md0
mdadm --assemble /dev/md0 /dev/vda5 /dev/dm-0
mount /dev/md0 /mnt
mount /dev/nasa/home /mnt/root
```

Update `mdadm.conf` and `chroot`

```
mdadm --detail --scan >> /mnt/etc/mdadm.conf
arch-chroot /mnt
```

Regenerate image, edit boot loader settings

Install some packages for later use:

```
pacman -Sy vim lvm2
```

Edit `/etc/mkinitcpio.conf` like this:

```
...
MODULES=(dm-raid raid0 raid1)
...
HOOKS=(base udev autodetect modconf block lvm2 mdadm_udev filesystems keyboard fsck)
...
```

In the `HOOKS` section, only `lvm2` and `mdadm_udev` are new.

Regereate image:

```
mkinitcpio -P
```

Edit `/etc/default/grub`:

```
...
GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet raid0.default_layout=2 root=/dev/md0"
...
```

Regenerate grub config file

```
grub-mkconfig -o /boot/grub/grub.cfg
```

Exit and reboot

```
exit
umount -R /mnt
reboot
```

2.2.2

Check the status of `/dev/md1`:

```
mdadm --detail /dev/md1
```

The array is missing a drive, so we add the unused drive to it:

```
mdadm --add /dev/md1 /dev/vda7
```

2.3

> Refs:
>
> https://gist.github.com/vodik/5660494
> https://wiki.archlinux.org/title/pacman#Installing_packages

When you run `pacman -Sy <package>`, `pacman` only looks at `<package>` and its dependencies, then update them. Old libraries are not kept. The problem is that other apps/libraries on the system may have common dependencies, but they are not updated when using `pacman -Sy <package>`. If the updated dependencies become too new for other apps/libraries to use (e.g. new api), things go really bad.

To avoid this, use `pacman -Syu <package>` instead. `-u` will do a full upgrade. ~~Or use other distros.~~