SA

Short Answer Questions

1. https://www.geeksforgeeks.org/what-happens-when-we-turn-on-computer/
http://www.tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/bootup.html

- computer power on
- CPU jumps to BIOS
- BIOS runs POST(power on self test, check hardware availability)
- BIOS Finds first bootable device

  BIOS will have devices available through POST. It will select the first boot device and gives back to CPU.
- MBR loading

  Including Primary boot loader code, which provides boot loader information and location details of actual boot loader code on the hard disk. This is helpful for CPU to load second stage of Boot loader. Partition table information, and Magic number which is validation check for MBR.
- GRUB loading

  GRUB loads the user-selected or default kernel into memory and passes control on to the kernel. If user do not select the OS, after a defined timeout GRUB will load the default kernel in the memory for starting it.
- Kernel

  Once Kernel loaded into to RAM, it always resides on RAM until the machine is shutdown. The first thing for kernel is executing INIT process.
- INIT

  It checks all the system properties, hardware, display, load kernel modules, file system check, file system mounting.
- User prompt

  It starts multiple instances of "getty" which waits for user logins.

2. https://www.howtogeek.com/56958/htg-explains-how-uefi-will-replace-the-bios/
https://www.maketecheasier.com/differences-between-uefi-and-bios/

   Since BIOS has been in use since the very beginning of computer industry, it works in 16-bit mode, limiting the amount of code that can be read and executed from the firmware ROM. UEFI stores all the information about initialization and startup in an .efi file instead of the firmware. This file is stored in EFI System Partition (ESP). The ESP partition will also contain the boot loader programs for the Operating System installed on the computer. Since UEFI is platform independent, it enhances the boot time and speed of the computer. The biggest

benefit of UEFI is its security over BIOS. UEFI can allow only authentic drivers and services to load, making sure that no malware can be loaded at computer startup. Secure Boot works by requiring a digital signature of boot loaders which should require digital signature by the Kernel. This process continues until the operating system is completely started.

3. https://linuxcluster.wordpress.com/2012/10/18/a-brief-look-at-the-difference-between-nfsv3-and-nfsv4/
   https://en.wikipedia.org/wiki/Network_File_System

- NFSv2 use UDP only, NFSv3 support UDP and TCP, NFSv4 use TCP only.
- NFSv2 only allows the first 2 GB of a file to be read due to 32-bit limitations. NFSv3 support for 64-bit file sizes and offsets, to handle files larger than 2 GB.
- NFSv2 limited the data transfer size to 8 KB. NFSv3 removed that limitation and allows the client and server to negotiate a maximum transfer size.
- NFSv3 supports for asynchronous writes on the server, improving write performance; additional file attributes in many replies, and avoid the need to re-fetch them than NFSv2.
- NFSv4 includes performance improvements, mandates strong security, the ability to provide scalable parallel access to files distributed among multiple server.
  Compatibility with firewall & NAT devices
  It has good performance on low-bandwidth connections
  Support unicode filenames

4. https://searchnetworking.techtarget.com/definition/Preboot-Execution-Environment
   https://en.wikipedia.org/wiki/Preboot_Execution_Environment

- Client machine is booted.
- The Network Interface Card of the machine triggers a DHCP request.
- DHCP server intercepts the request and responds with IP, subnet mask, gateway, DNS. In addition, it provides information about the location of a TFTP server and boot image (pxelinux.0).
- The client contacts the TFTP server for obtaining the boot image.
- TFTP server sends the boot image (pxelinux.0), and the client executes it.
  The boot image searches the pxelinux.cfg directory on TFTP server for boot configuration files on the TFTP server.
- The client downloads all the files it needs (kernel and root file system), and then loads them.
- Target Machine reboots.

Network Ninja

- Install arch linux

http://www.linuxandubuntu.com/home/a-step-by-step-guide-to-install-arch-linux

Download arch linux iso.

Do partition

cgdisk /dev/sda

set sda1 ef02 (BIOS boot partition)

set sda2 8300 (linux filesystem)

set sda3 8200 (linux swap)

lsblk (to check my partition)

mkfs.ext4 /dev/sda2

mkswap /dev/sda3

swapon /dev/sda3

mount /dev/sda2 /mnt

pacstrap /mnt base base-devel

genfstab –p /mnt > /mnt/etc/fstab

arch-chroot /mnt /bin/bash

vi /etc/locale.gen

unmark en_US.UTF-8 UTF-8

unmark zh_TW.UTF-8 UTF-8

locale-gen

echo " LANG=en_US.UTF-8" > /etc/locale.conf

ln –s /usr/share/zoneinfo/Asia/Taipei /etc/localtime

hwclock –systohc –utc

echo "$hostname" > /etc/hostname

systemctl enable dhcpcd.service

passwd (set root password)

useradd –m $user (add user if needed)

passwd $user (set user's password)

pacman –S intel-ucode (for intel cpu to boot up)

pacman –S grub

vi /etc/default/grub (Add GRUB_DISABLE_SUBMENU=y)

grub-mkconfig –o /boot/grub/grub.cfg

grub-install –target=i386-pc –recheck /dev/sda

exit

umount –R /mnt

reboot

set up nftables

https://home.regit.org/tag/ulogd/
https://wiki.archlinux.org/index.php/Nftables
http://computer-outlines.over-blog.com/article-nftables-7-nftables-logging-123303629.html
https://www.linux-dev.org/2012/08/logging-packages-with-iptables-and-ulog/
set up filter to log all packet sent from my vm
iptables -t filter -A OUTPUT '!' -o lo -j NFLOG --nflog-group 10400
ip6tables -t filter -A OUTPUT '!' -o lo -j NFLOG --nflog-group 10400
pacman –S ulogd (install ulogd)
vi ulogd.conf (to modify ulogd.conf to the following)
{
[global]
logfile="syslog"
plugin="/usr/lib/ulogd/ulogd_inppkt_NFLOG.so"
plugin="/usr/lib/ulogd/ulogd_raw2packet_BASE.so"
plugin="/usr/lib/ulogd/ulogd_filter_IFINDEX.so"
plugin="/usr/lib/ulogd/ulogd_filter_IP2STR.so"
plugin="/usr/lib/ulogd/ulogd_filter_PRINTPKT.so"
plugin="/usr/lib/ulogd/ulogd_output_LOGEMU.so"
stack=sal:NFLOG,sab:BASE,saif:IFINDEX,saip:IP2STR,sap:PRINTPKT,sao:LOGEMU
[sal]
group=10400
[sao]
file="/var/log/packet.log " (packet.log is made by me)
sync=1
}
ulogd -c ulogd.conf
ulogd –d
then every packet sent from my VM will be logged.

PXE&NFS

https://wiki.archlinux.org/index.php/PXE#Installation
https://wiki.archlinux.org/index.php/NFS

- I use virtual box and install virtualbox extension pack to support Intel interface card. In virtual box setting, I set all the three VM in NAT network.
- 3.1

→install arch linux in vm1

Download arch linux iso.
Do partition (1 part for boot, 1 part for normal data, 1 part for nfs later)

cgdisk /dev/sda
set sda1 ef02 (BIOS boot partition)
set sda2 8300 (linux filesystem)
set sda3 8300 (linux filesystem for nfs later)
lsblk (to check my partition)
mkfs.ext4 /dev/sda2 (make file system)
mkfs.ext4 /dev/sda3
mkdir /nfs
mount /dev/sda2 /mnt
mount /dev/sda3 /nfs (mount one partition at /nfs)
pacstrap /mnt base base-devel
genfstab –p /mnt > /mnt/etc/fstab
arch-chroot /mnt /bin/bash
vi /etc/locale.gen
unmark en_US.UTF-8 UTF-8
unmark zh_TW.UTF-8 UTF-8
locale-gen
echo " LANG=en_US.UTF-8" > /etc/locale.conf
ln –s /usr/share/zoneinfo/Asia/Taipei /etc/localtime
hwclock --systohc --utc
echo "$hostname" > /etc/hostname
systemctl enable dhcpcd.service
passwd (set root password)
pacman –S intel-ucode (for intel cpu to boot up)
pacman –S grub (for booting)
vi /etc/default/grub (Add GRUB_DISABLE_SUBMENU=y)
grub-mkconfig –o /boot/grub/grub.cfg
grub-install –target=i386-pc –recheck /dev/sda
exit
umount –R /mnt
reboot
→set up PXE server.
(ready the packages I need later)
pacman –S wget
pacman –S nfs-utils
pacman –S dnsmasq
wget archlinux.cs.nctu.edu.tw/iso/2018.05.01/archlinux-2018.05.01-x86_64.iso (get arch linux iso)

mkdir –p /nfs/archiso
mount –o loop, ro archlinux-2018.05.01-x86_64.iso /nfs/archiso (mount iso at /nfs/archiso)
ip a (check interface's name, my name is enp0s3 so I will use it in the later process)
ip link set enp0s3 up
ip addr add 192.168.0.1/24 dev enp0s3 (set static ip)
vi /etc/dnsmasq.conf (add the following into /etc/dnsmasq.conf file)
{
port=0
interface=enp0s3
bind-interfaces
dhcp-range=192.168.0.50,192.168.0.150,12h
dhcp-boot=/arch/boot/syslinux/lpxelinux.0
dhcp-option-force=209,boot/syslinux/archiso.cfg
dhcp-option-force=210,/arch/
dhcp-option-force=66,192.168.0.1
enable-tftp
tftp-root=/nfs/archiso
}
systemctl start dnsmasq.service
→**set up NFS server**
vi /etc/exports
{
/nfs/archiso 192.168.0.0/24(ro,no_subtree_check)
}
exportfs –rav
systemctl start nfs-server.service

- 3.2

I have already installed virtualbox extension pack
Set vm2 and vm3 network NAT network and boot priority set internet first.
After boot up, choose boot with NFS server. Press tab and edit archiso_nfs_srv to archiso_nfs_srv=${pxeserver}:/nfs/archiso and press enter. It will reach arch linux install mode.

- 3.3

→At vm1
vi /etc/exports (add the following in vm1)
{
/nfs 192.168.0.0/24(rw,sync,no_root_squash)

}
exportfs –rav
systemctl restart nfs-server.service
→at vm2
mkdir /mnt/nfs
ip a (check their interfaces name, in my case, enp0s3)
ip link set enp0s3 up
ip addr add 192.168.0.2/24 dev enp0s3 (set ip for my machine)
mount –t nfs –o vers=4 192.168.0.1:/nfs /mnt/nfs

→at vm3
mkdir /mnt/nfs
ip a (check their interfaces name, in my case, enp0s3)
ip link set enp0s3 up
ip addr add 192.168.0.3/24 dev enp0s3 (set ip for my machine)
mount –t nfs –o vers=4 192.168.0.1:/nfs /mnt/nfs

- 3.4

Automount in vm2 vm3
vi /etc/fstab (add the following line)
{
192.168.0.1:/nfs /mnt/nfs nfs defaults 0 0
}
mount –a (mount it at the time)
auto installation
https://shirotech.com/linux/how-to-automate-arch-linux-installation
https://github.com/setkeh/Kickstart/blob/master/archlinux-kickstart
in nfs-server(vm1) I create two file prepare.sh and install.sh in /nfs
after client machine boot with pxe and choose nfs option:
ip addr add 192.168.0.2/24 dev enp0s3
mount –t nfs –o vers=4 192.168.0.1:/nfs /mnt/nfs (mount client to nfs server to get my script)
then ./prepare.sh, it will auto install

- prepare.sh

#!/bin/bash
cp install.sh ~
cd ~ && ./install.sh

- install.sh

#!/bin/bash

```
Ip addr del 192.168.0.2/24 dev enp0s3
parted /dev/sda mklabel gpt
parted /dev/sda mkpart primary 1mib 513mib
parted /dev/sda mkpart primary 1g 100%
mkfs.ext4 /dev/sda2
mount /dev/sda2 /mnt
parted /dev/sda set 1 boot on
mkfs.fat -F32 /dev/sda1
mkdir /mnt/boot
mount /dev/sda1 /mnt/boot
pacstrap /mnt base base-devel
genfstab –p /mnt > /mnt/etc/fstab
arch-chroot /mnt /bin/bash
arch-chroot echo "en_US.UTF-8 UTF-8
zh_TW.UTF-8 UTF-8">>/etc/locale.gen
arch-chroot locale-gen
arch-chroot echo “ LANG=en_US.UTF-8” > /etc/locale.conf
arch-chroot hwclock --systohc --utc
arch-chroot echo “auto” > /etc/hostname
arch-chroot enable dhcpcd.service
arch-chroot pacman –S intel-ucode (for intel cpu to boot up)
arch-chroot pacman –S intel-ucode
arch-chroot pacman –S grub (for booting)
arch-chroot echo "GRUB_DISABLE_SUBMENU=y" >> /etc/default/grub
arch-chroot grub-mkconfig –o /boot/grub/grub.cfg
arch-chroot grub-install –target=i386-pc –recheck /dev/sda
exit
umount –R /mnt
reboot
```