This script prepares a **central Ubuntu server** that will:

1. **Serve boot files via TFTP** (using tftp-hpa)
2. **Serve the Raspberry Pi's root filesystem via NFS**
3. Allow multiple Pis to boot from the network by setting up per-device directories (e.g., red1, red8) using their **serial numbers** and **MAC addresses**

1. Use directory `/mnt/netboot2/nfs` as tftp-root directory
Note: To change the root tftp directory [i.e., the directory from which the tftp server will server files] change in `/etc/default/tftp-hpa` file the `TFTP_DIRECTORY="..."` field

This tells tftp-hpa where to find files when a Pi requests them during network boot(bootloader file examples : start4.elf, kernel*.img, cmdline.txt).

2. Download the image of the client [use the latest raspian lite image]
```
$ cd /mnt/netboot2
$ wget -O raspios_lite_armhf_latest.img.xz
https://downloads.raspberrypi.org/raspios_lite_armhf_latest
$ unxz raspios_lite_armhf_latest.img.xz
```
Note : ubuntu server doesn’t require unzip as it downloads as .iso

3. Mount the boot and root partitions from the image
```
$ cd /mnt/netboot2
$ kpartx -a -v raspios_lite_armhf_latest.img
$ mkdir bootmnt
$ mkdir rootmnt
$ mount /dev/mapper/loop0p1 ./bootmnt/
$ mount /dev/mapper/loop0p2 ./rootmnt/
```

`kpartx` command reads the partition table from the image provided and creates device files for the partitions in /dev/mapper. In our case it will create a `/dev/mapper/loop2p1` and `/dev/mapper/loop2p2` loop devices with the /boot and / partitions of the image.
bootmnt and rootmnt are just helper directories

4. Get the serial and mac address of the client machine. Suppose we work on red1, with serial number `bfb94a46` and MAC `e4:5f:01:f6:07:87`
```
$ cd /mnt/netboot2
$ PI_SERIAL=bfb94a46
$ PI_MAC=e4:5f:01:f6:07:87
$ PI_NAME=red1
```

```
$ SERVER_IP=192.168.2.1
```

5. Copy the mounted image ....
```
$ cd /mnt/netboot2
$ mkdir ./nfs/${PI_NAME}
$ cp -a ./rootmnt/* ./nfs/${PI_NAME}
$ cp -a ./bootmnt/* ./nfs/${PI_NAME}/boot
```
The second `cp` command will give an error for the `issue.txt` and `overlays`, so `cd ./nfs/${PI_NAME}/boot`, delete those files [which are broken symbolic links now] and copy again from `/mnt/netboot2/bootmnt/`
So, run:
```
$ cd /mnt/netboot2/nfs/${PI_NAME}/boot
$ rm issue.txt overlays
$ cp /mnt/netboot2/bootmnt/issue.txt .
$ cp -r /mnt/netboot2/bootmnt/overlays .
```

Now the /mnt/netboot2/{bootmnt,rootmnt} are not needed anymore, you can unmount them.

6. Update the /etc/fstab file *on the client's image* in order to indicate how to initialize the filesystem when the client is going to boot
```
$ vi /mnt/netboot2/nfs/${PI_NAME}/etc/fstab
=> Add the line:
192.168.2.1:/mnt/netboot2/nfs/red1 /boot nfs defaults,vers=3 0 0
# Remember that "red1" is the PI_NAME in this example
```

7. Update the `cmdline.txt` file *on the client's image*. This file is for passing arguments to the Linux kernel. Replace the current `cmdline.txt` in `/mnt/netboot2/nfs/${PI_NAME}/boot/` with
```
$ cat /mnt/netboot2/nfs/${PI_NAME}/boot/cmdline.txt
        console=serial0,115200 console=tty1 root=/dev/nfs
nfsroot=192.168.2.1:/mnt/netboot2/nfs/red1,vers=3 rw ip=dhcp rootwait elevator=deadline
```

8. Update the `/` file *on the server*. This file contains an entry for each directory that can be exported to NFS clients. So, add the following:
```
```

```
$ vi /etc/exports
# Add the two following lines =>
/mnt/netboot2/nfs/red1 *(rw,sync,no_subtree_check,no_root_squash)
/mnt/netboot2/nfs/red1/boot *(rw,sync,no_subtree_check,no_root_squash)
```

9. Add a symbolic link in the tftp-root directory to the /boot folder of the corresponding PI

```
$ cd /mnt/netboot2/nfs # => This is the tftp-root directory
$ ln -s ${PI_NAME}/boot ${PI_SERIAL}
```

Remember that tftp server searches by default in a directory named as the serial number for the configuration files of the corresponding client.

10. Restart dhcp, tftp, and nfs server and GO!

Note 1:
In this guide we don't do anything about SSH configuration. This is probably something we'll have to do.

Note 2:
I also download and install the nfs server on the server via `# apt install nfs-kernel-server`.

Note 3:
For the red8 configuration I additionally did the following in step 5:

```
$ rm /mnt/netboot2/nfs/${PI_NAME}/boot/start4.elf
$ rm /mnt/netboot2/nfs/${PI_NAME}/boot/fixup4.dat
$ wget https://github.com/raspberrypi/rpi-firmware/raw/master/start4.elf -P
/mnt/netboot2/nfs/${PI_NAME}/boot/
$ wget https://github.com/raspberrypi/rpi-firmware/raw/master/fixup4.dat -P
/mnt/netboot2/nfs/${PI_NAME}/boot/
```

but I didn't do this for the red1 and it didn't seem to matter.

Component	Purpose
TFTP	Transfers boot files like `start4.elf`, `cmdline.txt`, etc.
NFS	Mounts the root filesystem over the network
PXE/Netboot	Booting over LAN without SD card
Serial number (bfb94a46)	Used to identify each Pi uniquely for TFTP
cmdline.txt	Tells the kernel how to boot (e.g., use NFS)
fstab	Mounts /boot via NFS inside the running OS
DHCP	Assigns IP and tells Pi where to find TFTP server

LINKS:
https://www.reddit.com/r/raspberry_pi/comments/l7bzq8/guide_pxe_booting_to_a_raspberry_pi_4/?rdt=58378
https://linuxhit.com/raspberry-pi-pxe-boot-netbooting-a-pi-4-without-an-sd-card/
https://github.com/garyexplains/examples/blob/master/How%20to%20network%20boot%20a%20Pi%204.md