# MicroMEC netboot



This is a simple guide to setup a MicroMEC lab for test purposes using inexpensive components.

The setup uses a Linux based laptop acting as a netboot server. The server provides IP addresses (via DHCP) and the necessary boot loads to all MicroMEC nodes.

The MicroMEC nodes are Raspberry Pi 3B+ and 4B units. The RPi 3B+ units will run openSUSE Tumbleweed, the 4B units will run Rasbian Buster.

The MicroMEC nodes have no internal storage. All files are stored on the netboot server.

## Prepare MicroMEC Netboot Server

Our MicroMEC netboot server is based on Debian. Basic requirements are:

- installed dnsmasq
- installed tgt
- disabled any other dhcp or tftp servers
- at least two independent network interfaces (NICs) are available

An Ethernet NIC is configured with a static IP address.

This NIC will be connected to a network switch.

All the MicroMEC nodes will connect to the same switch via Ethernet.

This Ethernet based setup mimics the fast fiber or 5G connections between MECs in a real life deployment.

Our netboot server static IP configuration is as simple as that:

```
# cat /etc/network/interfaces
auto eth0
iface eth0 inet static
  address 192.168.4.1
  gateway 192.168.1.138
  netmask 255.255.255.0
```

The gateway IP address above is a dynamic IP address for the other NIC. Traffic between the MicroMEC nodes and the Internet will be routed via that gateway.

dnsmasq acts as a DHCP and TFTP server to enable network booting of the MicroMEC nodes. The dnsmasq configuration looks like this:

```
# cat /etc/dnsmasq.conf
interface=eth0,lo
domain=micromec.lan
dhcp-range=192.168.4.10,192.168.4.250,255.255.255.0,12h
pxe-service=0,"Raspberry Pi Boot"
enable-tftp
tftp-root=/srv/tftpboot/rpi
```

### Rootfs for netboot

During the netboot the MicroMEC RPi 3B+ nodes will mount their root file systems from the netboot server via iscsi. RPi 4B nodes will mount the rootfs via nfs (the iscsi boot is currently unsupported).

We have separate documents detailing how to prepare the rootfs on the netboot server:

- RPi 3B+ rootfs over iscsi
- RPi 4B rootfs over iscsi

### Initial test of the MicroMEC netboot server

Connect an RPi 3B+ (without an SD card) to the switch. Monitor `/var/log/daemon.log` on the MicroMEC netboot server

```
# tail -f /var/log/daemon.log
May  2 18:24:38 localhost dnsmasq-dhcp[130385]: DHCPDISCOVER(eth0) b8:27:eb:f3:26:91
May  2 18:24:38 localhost dnsmasq-dhcp[130385]: DHCPOFFER(eth0) 192.168.4.143 b8:27:eb:f3:26:91
May  2 18:24:38 localhost dnsmasq-tftp[130385]: file /srv/tftpboot/bootcode.bin not found
```

With this we have verified that the RPi received an IP address, and dnsmasq tried to push a file to the RPi.

In the next section we will describe what else is required on the netboot server in order to boot RPi 3B+ and RPi 4B MicroMEC nodes.

# Raspberry Pi 3B+ Netboot

We identify 4 stages for the boot process. The information about the boot process is also valid for RPi 4B units.

**Note 1**

These stages are not necessarily matching the stages that are referred and can be found in other documentation describing network booting of Linux computers.

**Note 2**

On **older RPi 3B+** models netboot was **not enabled** by the default firmware. This official Raspberry Pi document helps to check and upgrade the Pi's firmware if needed.

## Stage 1 Bootcode

The RPi 3B+ looks up `bootcode.bin` or `bootsig.bin` from the tftp boot directory. Our `bootcode.bin` is stored in `/srv/tftpboot/rpi` as this is the `tftp-root` we configured for dnsmasq.

## Stage 2 U-Boot

RPi 3B+ will continue the boot process by looking up files in a subdirectory that is derived from its serial number.

The serial number of an RPi can be obtained either via `/proc/cpuinfo` or from `/sys/firmware/devicetree/base/serial-number`.

In our case the RPi 3B+ has this information:

```
$ cat /sys/firmware/devicetree/base/serial-number
0000000007f32691
```

The logs of dnsmasq will show the files that the RPi 3B+ will attempt to load. However in our case the next is `config.txt` that is relevant. Unlike PCs the RPi does not have a BIOS to store important configuration parameters. `config.txt` is used for this purpose. The structure and the use of this file is well described in this document.

Files for stage 3 can be placed in the `07f32691` subdirectory in `tftp-root`. In our case we store these files in `/srv/tftpboot/07f32691`.

This stage loads U-Boot. U-Boot is then loads the initramfs and the kernel in the next stage.

We have the following files and subdirs for the RPi 3B+ for stage 2:

Files:

- start.elf
- fixup.dat
- u-boot.bin
- bcm2710-rpi-3-b-plus.dtb
- Image-5.6.8-1-default
- initrd-5.6.8-1-default
- config.txt
- cmdline.txt
- ubootconfig.txt

Subdirs:

- overlays

These files can be found in the openSUSE Tumbleweed ARM JeOS Raspberry Pi 3 image that is available from OBS.

The raw image file of the RPi 3 can be mounted via a loop back device on a Linux computer:

```
# mkdir /tmp/rpi3_rootfs
# partx -a ./openSUSE-Tumbleweed-ARM-JeOS-raspberrypi3.aarch64-2020.03.25-Snapshot20200421.raw
# mount /dev/loop0p3 /tmp/rpi3_rootfs
```

After that the files for stage 2 can be located in `/tmp/rpi3_rootfs/boot/vc`.

The rootfs subdir which we have created under `/srv/tftpboot/07f32691` on the netboot server also comes from the above mentioned image. This is what the RPi 3B+ will mount in the last stage of the boot.

Alternatively OBS also produces an image with the rootfs only. This can also be used for our base rootfs if we wish to do so.

As a side note: having the rootfs mounted via a loop back device allows us to manipulate the content on our Linux workstation. With this method we can customize the rootfs of each of the MicroMEC nodes that are booted from the net.

### Stage 3 initramfs and kernel

During this stage the RPi loads a configuration which let us specify the location of the initramfs and the kernel. This configuration is stored in a file that is named based on the MAC address of the Ethernet NIC of the RPi.

In our case the RPi 3B+ has this config on the netboot server:

```
$ cat /srv/tftpboot/rpi/pxelinux.cfg/01-b8-27-eb-f3-26-91
menu title Linux selections

# wait 20 / 10 = 2 sec
timeout 10

default openSUSE_TW

label openSUSE_TW
  menu label openSUSE Tumbleweed RPi 3B+ Image
  initrd 07f32691_3b+_openSUSE/rootfs/boot/initrd
  kernel 07f32691_3b+_openSUSE/rootfs/boot/Image
```

We can have several initramfs and kernel combinations. We can select what to boot if we the UART console of the RPi is enabled and we have a suitable *USB to TTL adapter*.

The initramfs, the kernel and the rootfs are all built with Open Build Service (OBS). OBS is a free service provided by the openSUSE community. It uses kiwi to build various artifacts, such as ISO images, rootfs images, VM appliances or container images.

In the future we will try to build a custom MicroMEC image which will minimize the provisioning efforts.

We use the latest openSUSE Tumbleweed JeOS (Just enough Operating System) image for the RPI 3B+.

## Stage 4 mount and switch to rootfs

For the preparation of the rootfs on the netboot server please refer to the RPi 3B+ iscsi how-to document.

Once the initramfs is properly loaded and the necessary processes started the RPi 3B+ will attempt to mount and switch to the rootfs that resides on the netboot server. The *cmdline.txt* defines how the rootfs can be accessed.

In our case `cmdline.txt` has the following content:

```
$ cat /srv/tftpboot/07f32691/cmdline.txt
console=serial0,115200 loglevel=4 rd.shell ip=dhcp netroot=iscsi:192.168.4.1::::iqn.org.micromec:rpi3-1-rootfs rd.
iscsi.login_retry_max=10 root=UUID=7bf4dc05-cd4a-46af-9689-4a03209d5ed2 rootfstype=ext4 rw rootwait
```

This command line will instruct the RPi to look for the root filesystem over the network and mount it as an iscsi device. When the RPi mounts the storage device over the network it will appear as a "local" device (like the microSD card used to be). When the rootfs switching stage is reached the RPi will use that virtual device. In the `cmdline.txt` above we refer to that device by its UUID.

**Note**

The UUID of the root device can be determined by login into the iscsi target on your workstation first as described in the *Remote Testing* section of the RPi 3B+ rootfs over iscsi document.

Just for a quick recap on how to login to an iscsi target:

1. Discover the iscsi target

   ```
   $ sudo iscsiadm --mode discovery --op update --type sendtargets --portal bootserv
   192.168.4.1:3260,1 iqn.org.micromec:rpi3-1-opensuse-rootfs
   ```

2. Login to the iscsi target

   ```
   $ sudo iscsiadm -m node --targetname iqn.org.micromec:rpi3-1-opensuse-rootfs -p bootserv -l
   Logging in to [iface: default, target: iqn.org.micromec:rpi3-1-opensuse-rootfs, portal: 192.168.4.1,3260]
   Login to [iface: default, target: iqn.org.micromec:rpi3-1-opensuse-rootfs, portal: 192.168.4.1,3260]
   successful.
   ```

3. Check the available partitions

   ```
   $ cat /proc/partitions
   major minor #blocks name ..... 8 16 4096000 sdb
   ```

4. Determine the UUID

```
$ sudo blkid /dev/sdb
/dev/sdc: UUID="7bf4dc05-cd4a-46af-9689-4a03209d5ed2" BLOCK_SIZE="4096" TYPE="ext4"
```

Details on possible cmdline entries can be found here:

http://man7.org/linux/man-pages/man7/dracut.cmdline.7.html

Detailed information on how to enable netboot for an RPi 3B+:

https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net.md

https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net_tutorial.md

https://metebalci.com/blog/bare-metal-rpi3-network-boot/

## Raspberry Pi 4B Netboot

Older RPi 4Bs might require an EEPROM update to enable netboot.

Let's check our RPi 4Bs while still having Raspbian booted from the SD card. Get the version of the bootloader like that:

Unit #1:

```
# vcgencmd bootloader_version
May 10 2019 19:40:36
version d2402c53cdeb0f072ff05d52987b1b6b6d474691 (release)
timestamp 0
```

Unit #2:

```
# vcgencmd bootloader_version
Sep 10 2019 10:41:50
version f626c772b15ba1b7e0532a8d50a761b3ccbdf3bb (release)
timestamp 0
```

If the output shows "Sep 10 2019" or an earlier date, then the RPi 4B needs a new firmware to make net booting possible. In our case both RPis need a new bootloader.

Follow https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711_bootloader_config.md or

https://kiljan.org/2019/11/16/arch-linux-arm-network-boot-on-raspberry-pi-4/ or

https://hackaday.com/2019/11/11/network-booting-the-pi-4/ for the firmware update.

Completing the firmware update the response should look something like this:

```
# vcgencmd bootloader_version
Apr 16 2020 18:11:26
version a5e1b95f320810c69441557c5f5f0a7f2460dfb8 (release)
timestamp 1587057086
```

Similarly to the RPi 3B+ the RPi 4B will also try to look up files from a subdir derived from the serial number. Please refer to the *Raspberry Pi 3B+ Netboot* section on how to determine the name of the subdir.

In our case the RPi 4B unit #1 looked up the `0dc0a15d` subdir and unit #2 looked up the `1aed06b2` subdir.

The boot process and the requirements for the netboot server are basically the same for RPi 4B as we already describe for the RPi 3B+. The differences are the files that are needed for the RPi 4B.

The RPi 4B specific files can be extracted from an official Rasbian image for RPi 4B. We have used the Raspbian Buster Lite image.

Files that are needed for stage 2 are:

- start4.elf
- fixup4.dat
- kernel.8
- bcm2711-rpi-4-b.dtb
- config.txt
- cmdline.txt

Subdirs:

- overlays

The procedure to copy the relevant files is very similar to the one we had for RPi 3B+ above. For RPi 4B we have to copy the files from the Rasbian Buster lite image.

For the preparation of the rootfs on the netboot server please refer to RPi 4B iscsi howto document.

The RPi 4B iscsi howto document explains how to mount the Raspbian Buster lite image. Once the image is mounted we can find the files in the `boot` subdirectory.

Once the files are copied, we can adjust the `cmdline.txt` file to something like this

```
$ cat cmdline.txt
dwc_otg.lpm_enable=0 console=serial0,115200 loglevel=7 modules=iscsi_tcp ip=dhcp netroot=iscsi:192.168.4.1::::iqn.
org.micromec:rpi4-1-raspbian-rootfs rd.iscsi.login_retry_max=10 root=UUID=8236bee6-9c37-4b04-8092-2630fd2b0596
rootfstype=ext4 rw rootwait
```

**Note**

The UUID of the root device can be determined by login into the iscsi target on your workstation first as described in the *Remote Testing* section of the RPi 4B rootfs over iscsi document.

Just for a quick recap on how to login to an iscsi target:

1. Discover the iscsi target

   ```
   $ sudo iscsiadm --mode discovery --op update --type sendtargets --portal bootserv
   192.168.4.1:3260,1 iqn.org.micromec:rpi3-1-opensuse-rootfs
   192.168.4.1:3260,1 iqn.org.micromec:rpi4-1-opensuse-rootfs
   192.168.4.1:3260,1 iqn.org.micromec:rpi4-1-raspbian-rootfs
   ```

2. Login to the iscsi target

   ```
   $ sudo iscsiadm -m node --targetname iqn.org.micromec:rpi4-1-raspbian-rootfs -p bootserv -l
   Logging in to [iface: default, target: iqn.org.micromec:rpi4-1-raspbian-rootfs, portal: 192.168.4.1,3260]
   Login to [iface: default, target: iqn.org.micromec:rpi4-1-raspbian-rootfs, portal: 192.168.4.1,3260]
   successful.
   ```

3. Check the available partitions

   ```
   $ cat /proc/partitions
   major minor #blocks name ..... 8 16 4096000 sdc
   ```

4. Determine the UUID

   ```
   $ sudo blkid /dev/sdc /dev/sdc: UUID="8236bee6-9c37-4b04-8092-2630fd2b0596" BLOCK_SIZE="4096" TYPE="ext4"
   ```