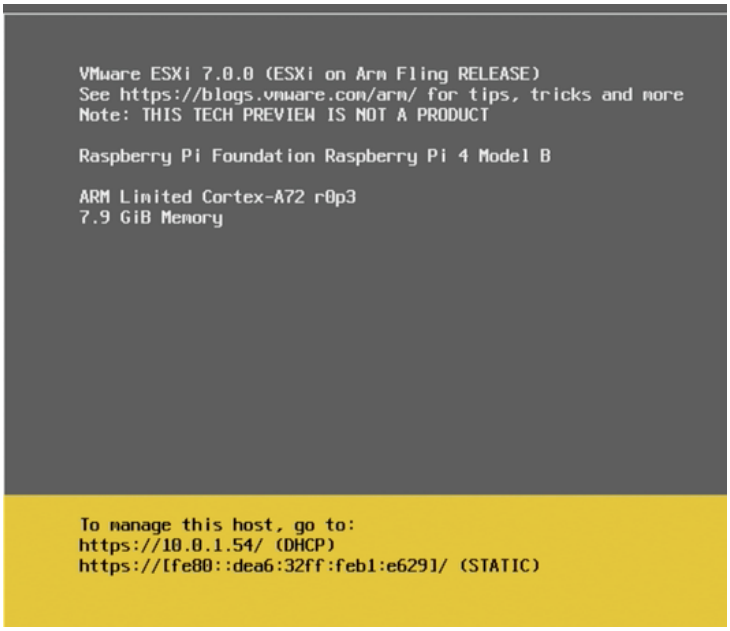
Fling on Raspberry Pi



1. Info

The Raspberry Pi 4B is a credit-card size single board computer (SBC). It is based on the Broadcom BCM2711 SoC with 4 x Cortex-A72 and supports up to 8GiB RAM. The SBC also has USB3 and GbE connectivity.

See https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

The anticipated use case is "Far Edge": e.g. a virtualized IoT gateway.

2. Required and supported hardware

Minimally, you need:

- A good power supply
- Raspberry Pi 4 Model B
    - 4GiB or 8GiB (1GiB and 2GiB are not supported)
    - An SD card (for UEFI firmware)
- 1 x micro SD card for UEFI firmware
- 1 x USB drive for installer ISO
- 1 x USB drive (or SATA / NVMe adapter) for the actual ESXi installation
- Console selection
    - HDMI + USB keyboard
    - UART (serial) cable

The following hardware is supported:

- On-board GbE NIC (recommended)
- USB storage
- USB keyboard
- USB networking
- PoE HAT
- HDMI video
- Serial console

As you will note, SD card is not supported. It is only used to keep the UEFI firmware.

2.1. Power Supply

A good power supply is critical, especially for using USB storage.

Description	Amps	Product
Argon ONE Raspberry Pi 4 USB Type C Cable Power Supply	3.5	https://www.amazon.com/gp/product/B07TW4Q693
CanaKit 3.5A Raspberry Pi 4 Power Supply	3.5	https://www.amazon.com/CanaKit-Raspberry-Power-Supply-USB-C/dp/B07TYQRXTK

2.2. Cooling

It is highly recommended to perform active cooling. The Pi runs hot. Passive cooling (heatsinks) help here, but so will a nice quiet 5V Noctua fan. Cooling and cases are highly individual and may involve some handywork.

2.3. HDMI

Use a micro-HDMI adapter or cable like https://www.amazon.com/CanaKit-Raspberry-Micro-HDMI-Cable/dp/B07TTKD38N.

2.4. HATs

2.4.1. PoE

These can be used for building clusters of Pies. Note that it is highly recommended to provide additional active cooling. PoE HATs make an already hot Pi run even hotter, without any space for passive cooling elements and (at best) puny fans.

Description	Product	Comments
GeeekPi Raspberry Pi 4 PoE HAT	https://www.amazon.com/gp/product/B0833PP65P	Fan works. Largest fan so far seen on a PoE hat and there is space for heatsinks.
UCTRONICS PoE HAT for Raspberry Pi 4	https://www.amazon.com/UCTRONICS-Raspberry-Ethernet-Expansion-Cooling/dp/B082ZJYCS3	Fan works. No space for heatsinks.

Official Raspberry Pi Power Over Ethernet	https://www.amazon.com/poe-hat/dp/B07GR9XQJH	Fan doesn't spin due to missing UEFI support. No space for heatsinks. **Must use external cooling!**

2.4.2. Other HATs

Not supported. This includes any kind of I2C, SPI or GPIO extension or connectivity.

2.5. Optional Serial Console

ESXi-Arm on Pi is entirely usable via HDMI + USB keyboard, yet for developers and power users alike, the importance of a serial connection cannot be overstated. In ESXi it gives you convenient access to system log, console and basic management interface (DCUI), especially if you chose to operate your Pi headless.

Note: Get a USB-to-TTL serial cable. The cable must be for 3.3V, not 5V.

VID	PID	Description	Product	Comments
0403	6001	DTECH FTDI USB to TTL Serial 3.3V Adapter Cable	https://www.amazon.com/dp/product/B07RBKCW3S	

2.6. USB devices

IMPORTANT: USB device can consume significant power and thus put a stress on Pi's power circuits. Some USB devices can consume so much power (e.g. NVMe enclosure) that the Pi will simply not work, be unstable or have unstable USB behavior. For anything short of a basic USB key, use a powered USB3 hub.

Note: Some of these USB devices have Type-C plugs. The expectation is that they will be plugged (directly or via hub) into the front (USB3) ports. Use a mechanical adapter such as https://www.amazon.com/gp/product/B07LF72431. Do not plug these into the Type-C port on the Pi for performance reasons.

See the lists and notes in the main ESXi-Arm Fling Doc.

3. Preparation

Setup involves ensuring Raspberry Pi microcode (e.g. for USB) is up-to-date, deploying the UEFI firmware and preparing the installer USB drive.

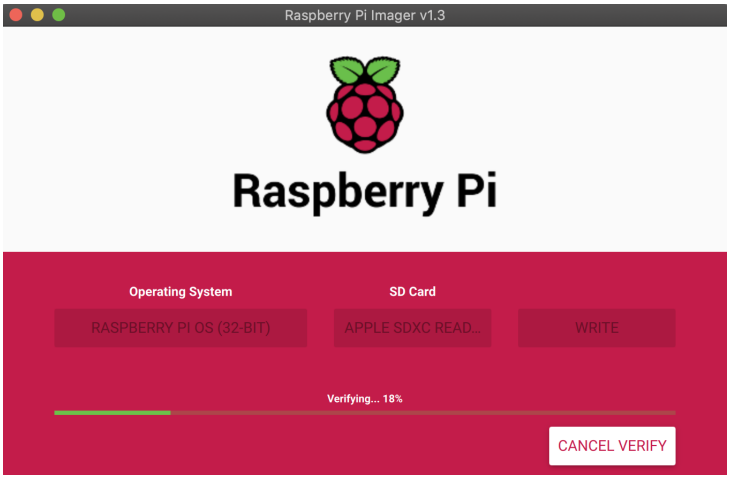
3.1. Ensure Raspberry Pi EEPROM is updated

This is critical to a stable USB experience and reasonable temperatures.

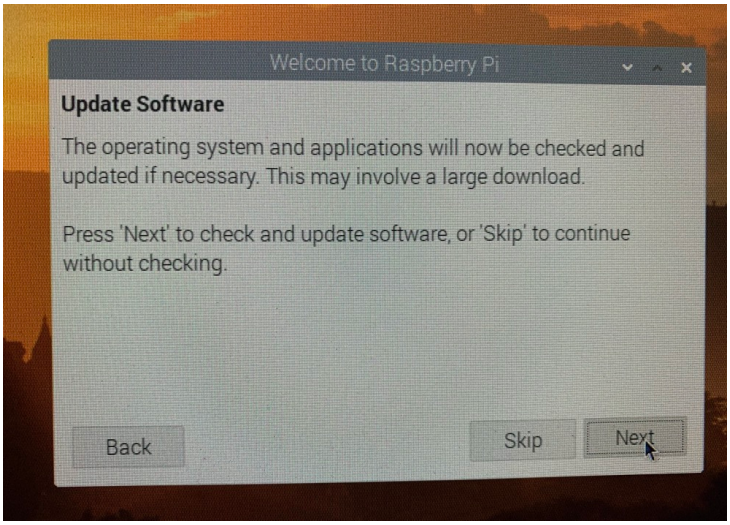
You need:

- SD Card
- Mouse and Keyboard
- HDMI Screen
- Raspberry Pi Imager Tool for your OS: https://www.raspberrypi.org/downloads/

Deploy the Pi OS onto the SD card (If card isn't formatted, you can use Imager Tool to format): (Note if you purchased the Pi that came along with a pre-imaged SD card with Pi OS, go ahead and skip to the sudo rpi-eeprom step below)

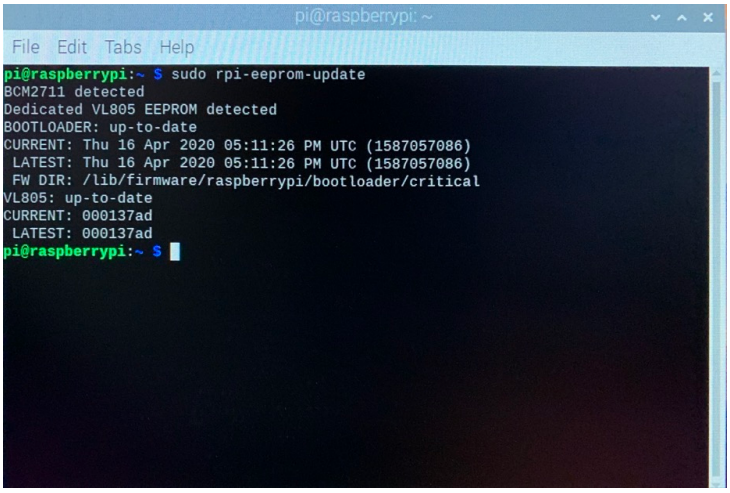


Plug SD card into the Pi and power up device with network, keyboard and mouse connected. Complete the setup wizard and check for update and then reboot.



After Pi has rebooted, open the terminal and run the following command to verify you've got the latest EEPROM.

```
$ sudo rpi-eeprom-update
```



If the tool reported any available updates, apply them:

```
$ sudo rpi-eeprom-update -a
$ sudo reboot
```

```
pi@raspberrypi:~ $ sudo rpi-eeprom-update
BCM2711 detected
VL805 firmware in bootloader EEPROM
*** UPDATE AVAILABLE ***
BOOTLOADER: update available
CURRENT: Thu 19 Mar 14:27:25 UTC 2020 (1584628045)
 LATEST: Thu 16 Apr 17:11:26 UTC 2020 (1587057086)
 FW DIR: /lib/firmware/raspberrypi/bootloader/critical
VL805: up-to-date
CURRENT: 000137ad
 LATEST: 000137ad
pi@raspberrypi:~ $ sudo rpi-eeprom-update -a
BCM2711 detected
VL805 firmware in bootloader EEPROM
BOOTFS /boot
*** INSTALLING EEPROM UPDATES ***
BOOTLOADER: update available
CURRENT: Thu 19 Mar 14:27:25 UTC 2020 (1584628045)
 LATEST: Thu 16 Apr 17:11:26 UTC 2020 (1587057086)
 FW DIR: /lib/firmware/raspberrypi/bootloader/critical
VL805: up-to-date
CURRENT: 000137ad
 LATEST: 000137ad
BOOTFS /boot
EEPROM updates pending. Please reboot to apply the update.
pi@raspberrypi:~ $ _
```

Otherwise, shutdown the Pi :

```
$ sudo shutdown
```

3.2. Setup UEFI on SD Card

The SD card will be only used for UEFI firmware, so don't bother with a big card. The SD card is required, and configurations where the UEFI firmware is booted from USB or network are not covered here or supported.

3.2.1. Download the necessary bits

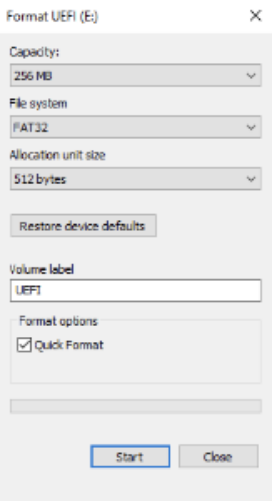
- Download the latest official Raspberry Pi Firmware and extract the contents to your computer, you should have a folder called **firmware-master**. This corresponds to the microcode necessary to initialize the Raspberry Pi.
- Download the latest community Raspberry Pi 4 UEFI firmware and extract the contents to your compute you should have a folder called **RPi4_UEFI_Firmware_v1.20**. This is the firmware necessary to boot ESXi-Arm.

3.2.2. Prepare SD card.

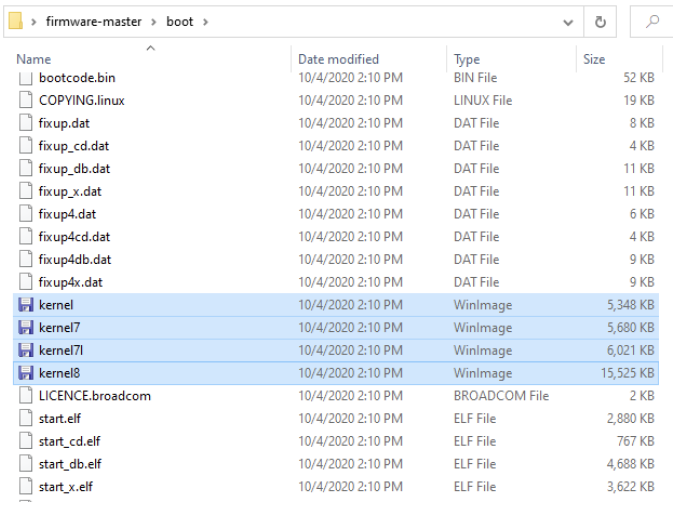
Format the SD card with a single FAT32 (MSDOS) partition.

3.2.2.1. On Widows

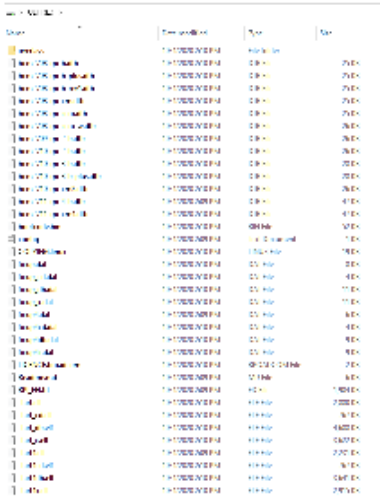
Open up Windows Explorer and identify the SD Card and select "Format" and create FAT32 partition. In the example below, the partition label is called UEFI:



Extract firmware-master.zip and then delete all files starting with **kernel*.img** within **firmware-master/boot** directory and then copy the entire "boot" directory onto the newly formatted SD card:



Extract RPi4_UEFI_Firmware_v1.20.zip and copy all files within the **RPi4_UEFI_Firmware_v1.20** directory into the same boot directory (confirm override of files when prompted) on SD card:



4GB Pi 4 only: Edit the **config.txt** file on the SD Card and append **gpu_mem=16**:

Eject the SD Card and then put the SD card into the Pi.

3.2.2.2. On macOS

Identify the disk using the following command and make note of the disk path (e.g. /dev/diskX):

```
$ diskutil list

/dev/disk6 (internal, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *63.9 GB    disk6
   1:             Windows_FAT_32 boot                    268.4 MB   disk6s1
   2:                      Linux                         63.6 GB    disk6s2
```

Create FAT32 partition on the SD Card by running the following command and providing disk path. The partition label will be called **UEFI**, you can choose another name if you wish:

```
$ diskutil partitionDisk /dev/disk6 1 MBRFormat "MS-DOS" UEFI R
```

Delete all files starting with **kernel*.img** within **firmware-master/boot** directory and then copy the entire "boot" directory onto the newly formatted SD card:

```
$ rm ~/Desktop/firmware-master/boot/kernel*.img
$ cp -rf ~/Desktop/firmware-master/boot/* /Volumes/UEFI
```

Copy all files within the **RPi4_UEFI_Firmware_v1.20** directory into the same boot directory on SD card:

```
cp -rf ~/Desktop/RPi4_UEFI_Firmware_v1.20/* /Volumes/UEFI
```

4GB Pi 4 only: Edit the **config.txt** file on the SD Card and append **gpu_mem=16**:

```
echo "gpu_mem=16" >> /Volumes/UEFI/config.txt
```

Eject the SD Card:

```
diskutil eject /dev/disk6
```

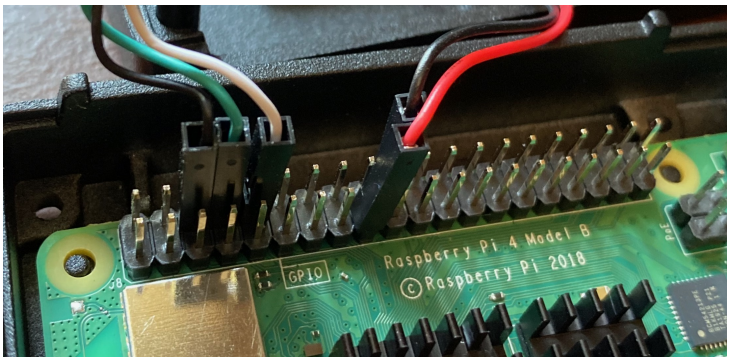
Now put the SD card into the Pi.

3.3. Decide on your console choice

The choices are:

- HDMI + USB keyboard (if you choose this option, skip to 3.4)
  - Required to update Raspberry Pi's microcode EEPROM via Pi OS
  - Access to UEFI setup
  - Preferred for installing ESXi-Arm
- Serial console
  - Access to UEFI setup
  - Supported for installing ESXi-Arm

If you chose to wire up the serial port, connect the cable to the UART pins on the board. The three pins you would need to connect are GND, TX and RX.



3.3.1. Wiring

Note the connections are as follows, with the TX pin on the cable going to the RXD and vice-verse. If you get this wrong, you will see no output.

```
Black (GND)  GND (Ground, pin 6)
Green (RX)   TXD (GPIO 14, pin 8)
White (TX)   RXD (GPIO 15, pin 10)
```

3.3.2. Terminal emulator

Fire up your terminal emulation and connect to the device on your PC. The parameters used to open this port:

```
Baud Rate     115200
Data Bits     8
Parity        None
Stop Bits     1
```

```
Flow Control  None
```

3.3.2.1. 'screen' terminal emulator

Note: device names below may be different. Check your system.

On Linux:

```
$ screen /dev/ttyUSB0 115200
```

On macOS:

```
$ screen /dev/tty.usbserial-A900UE2E
```

3.3.2.2. 'minicom' terminal emulator

Note: device names below may be different. Check your system.

With **minicom**, you will have to configure settings the first time you use it. To access menus, you will have to use the **CTRL** key in Linux, and **ESC** key on macOS. These directions will refer to this key as **META**.

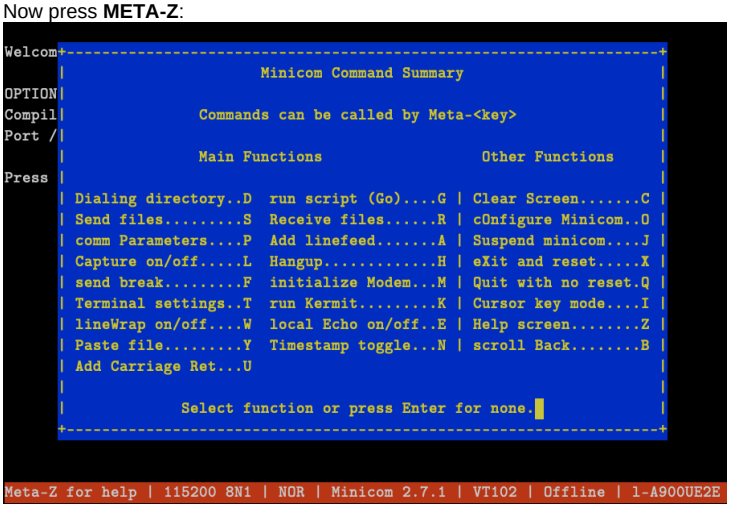
On Linux:

```
$ minicom -c on -D /dev/ttyUSB0
```

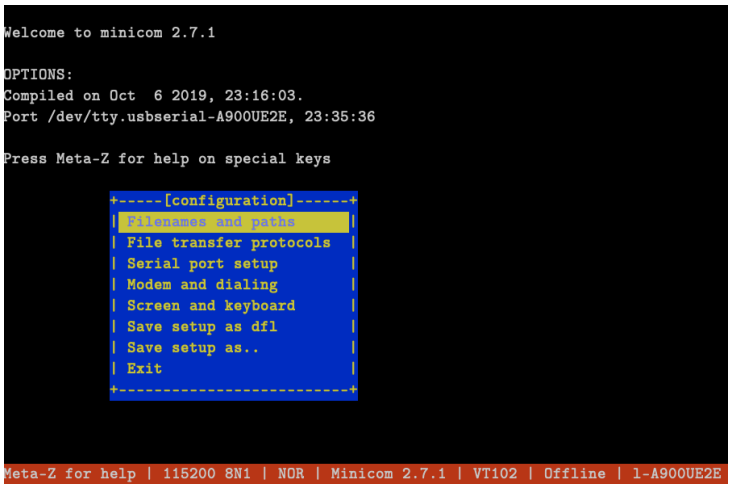
On macOS:

```
$ minicom -c on -D /dev/tty.usbserial-A900UE2E
```

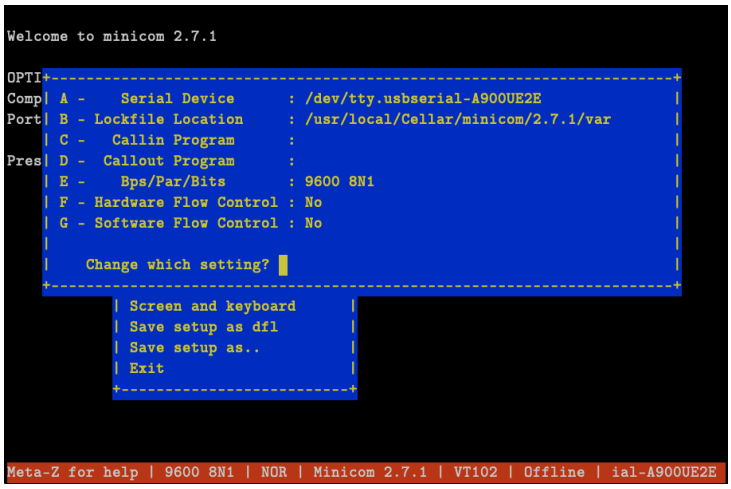
Now press **META-Z**:



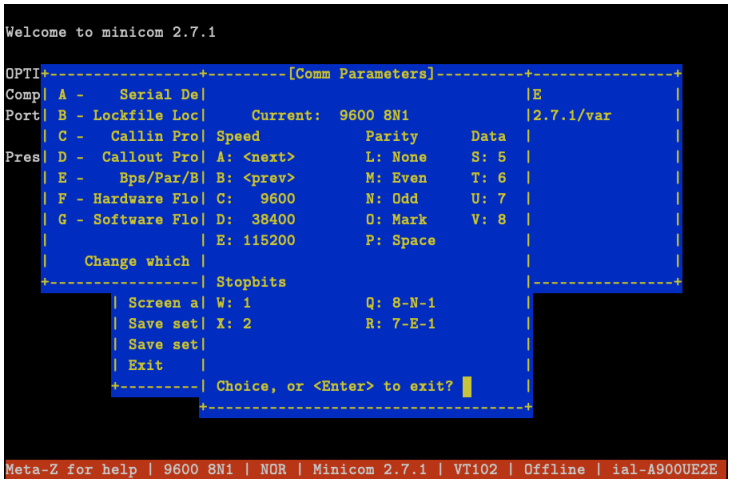
Now press **O**:



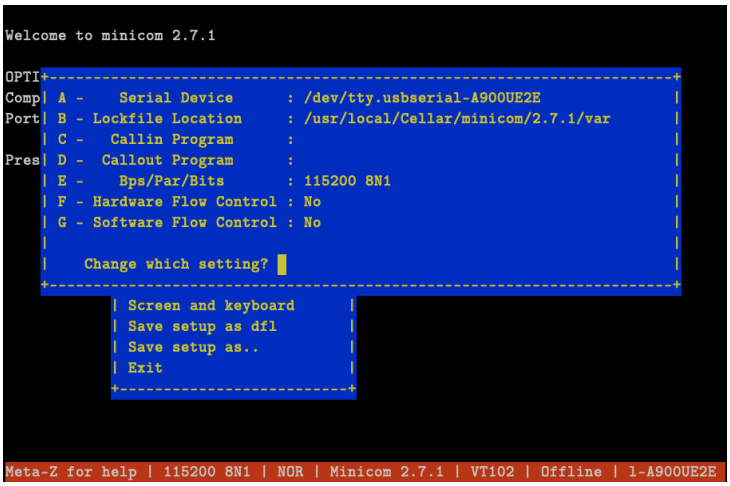
Use arrow key to navigate to **Serial port setup** and press the **ENTER**:



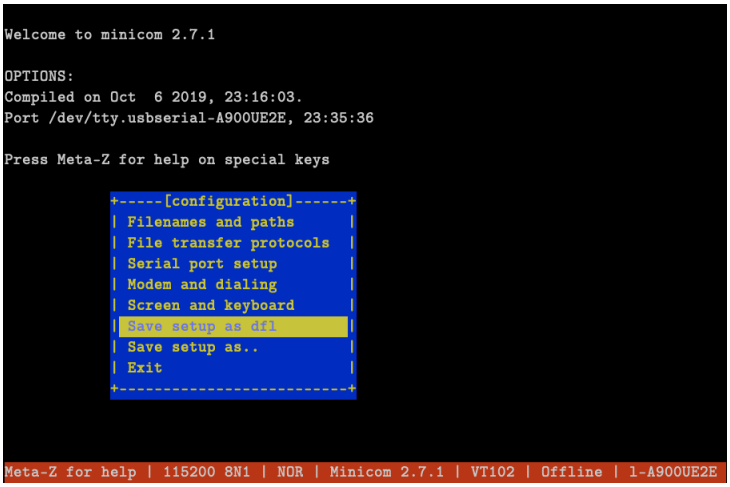
Now press **E**:



Press **E** again, then **ENTER**.



Make sure settings **F** and **G** both say **No** to any kind of flow control. Press **ENTER** when done, then navigate to **Save setup as dfl** and press **ENTER**.

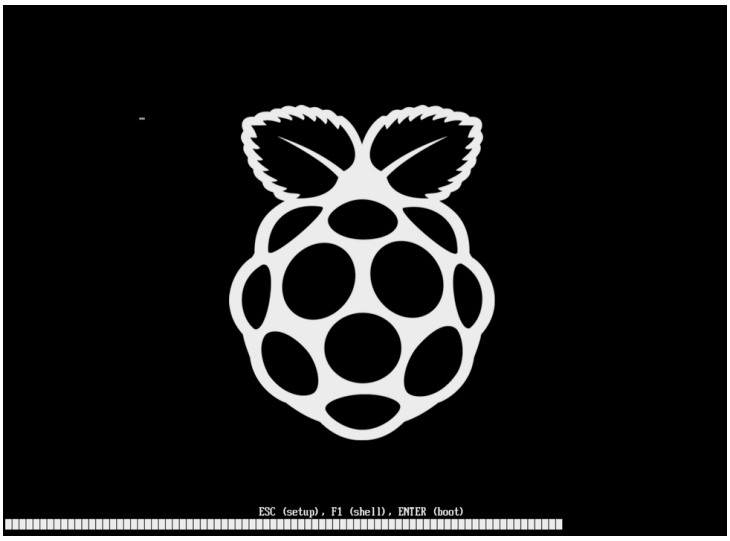


Use **ESC** to exit out of the menus.

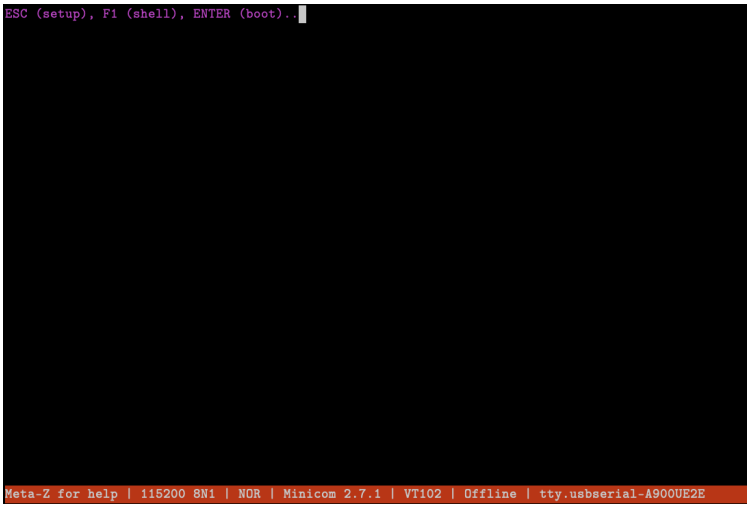
3.4. UEFI firmware configuration

You can use either HDMI + USB keyboard or a serial console. The output is duplicated to both. To enter UEFI setup, apply power to the Raspberry Pi, then keep mashing the **ESC** key when prompted.

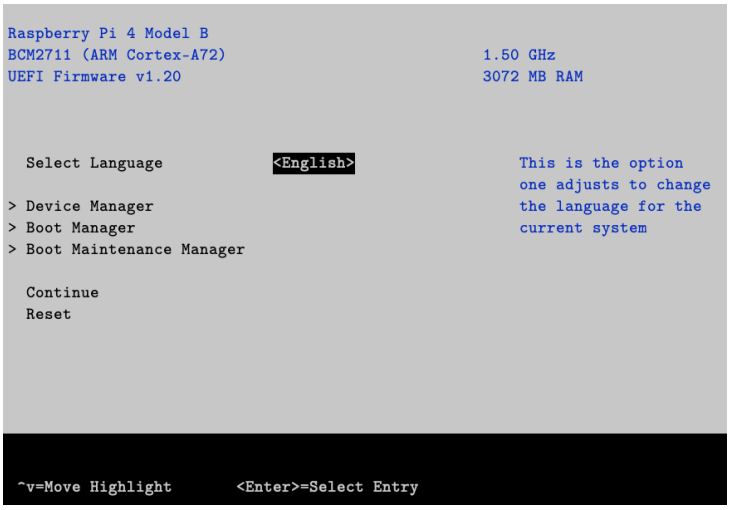
With an HDMI screen, this is when you see the Raspberry Pi logo.



With a serial console, this is when you see this:



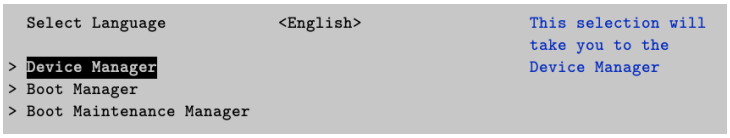
In either case, the output will be identical - the configuration home page:



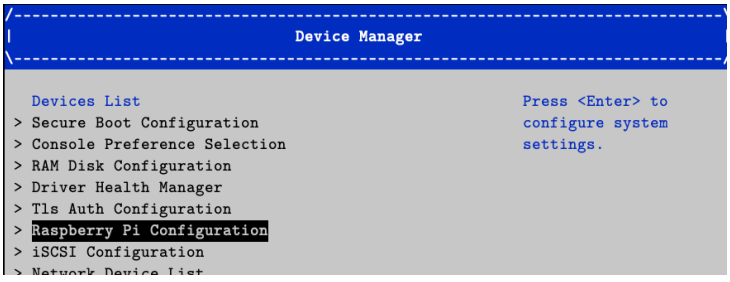
3.4.1. Disable 3GiB memory limit

The Raspberry Pi 4 UEFI is configured with a default limit of 3GiB of memory for OS compatibility purposes. This will prevent the ESXi installer from proceeding, and needs to be disabled.

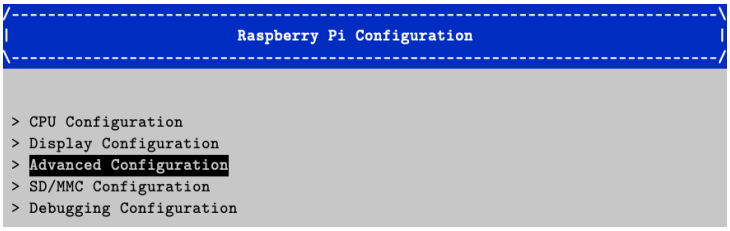
Using arrow keys, first navigate to **Device Manager:**



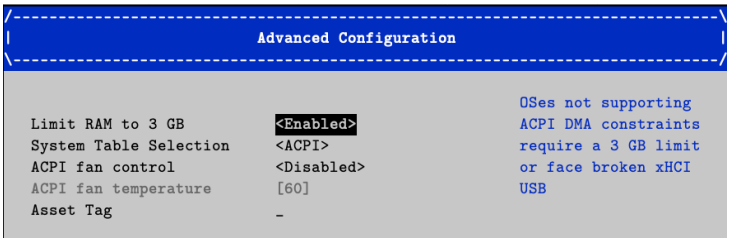
Press **ENTER** and navigate to **Raspberry Pi Configuration**:



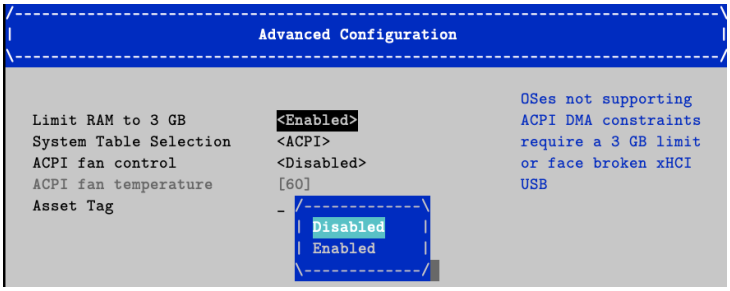
Press **ENTER** and navigate to **Advanced Configuration**:



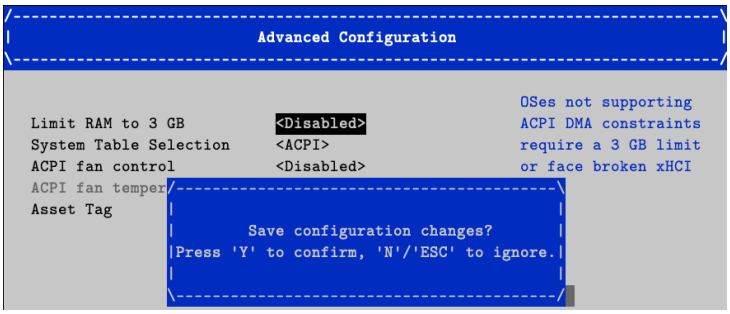
The **Limit RAM to 3GB** setting should already be selected, as it is the first setting on the page:



Press **ENTER** and use arrow keys to select **Disabled**:



Press **ENTER** again, then **F10** to save settings:



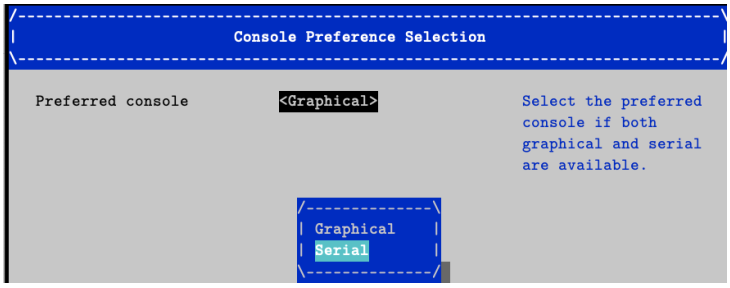
Press Y, then **ESC** three times to get back to the home page. Then navigate to **Continue** and press **ENTER.**

Press **ENTER** again. The Pi will reboot.

3.4.2. Console Preference Selection

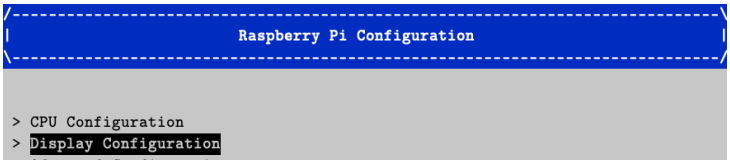
The console preference setting only matters if you have an HDMI screen connected to the Pi. If the screen is not connected, the serial port will be exposed to the booted OS, and ESXi will use it as it will detect a headless mode.

If the HDMI screen is connected, the serial port will not be exposed to the booted OS by default. Change the setting to Serial if you want to use the serial port in ESXi.

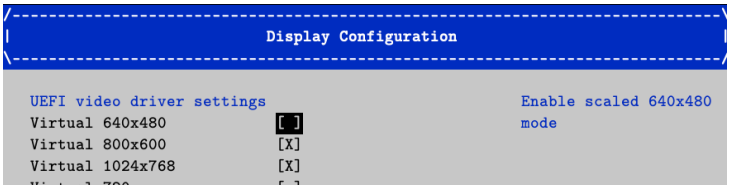


3.4.3. Raspberry Pi Display Configuration

The display settings can be used to force a particular resolution to improve legibility:



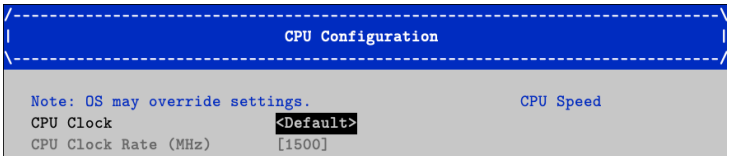
For example, with a small screen this may work well:



3.4.4. Raspberry Pi CPU Configuration

It is not recommended to mess with these. Regardless of the MHz reported, the Pi won't go over 1500 without additional settings in **config.txt** on the SD card.

Note: do not overclock. Not only are you compromising stability, but memory and I/O performance may suffer up to 2x.



4. Install ESXi-Arm

Follow the generic installation steps, with a few caveats.

On the Raspberry Pi, ESXi only supports installation to USB storage or iSCSI LUNs.

It is recommended to use HDMI video + USB for installation itself. If you need to pass any advanced options to installer (e.g. via **Shift-O** in the ESXi bootloader), this cannot be done using serial console today.

E.g. Append **autoPartitionOSDataSize=8192** for an 8GB VMFS-L partition, and the rest available for a datastore.

For more details on changing the default OSData volume, please see this blog post.

4.1. Power

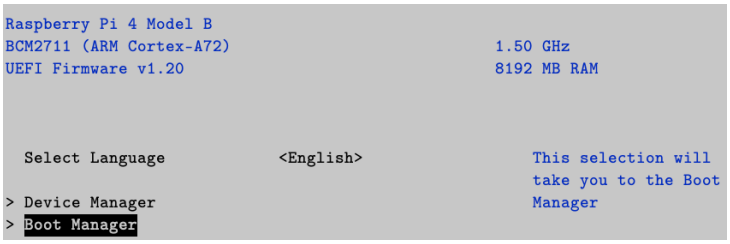
Make sure to use a solid power supply. If plugging in USB NICs or any storage beyond a very basic USB thumb drive, consider using a powered USB hub.

4.2. Automated installation

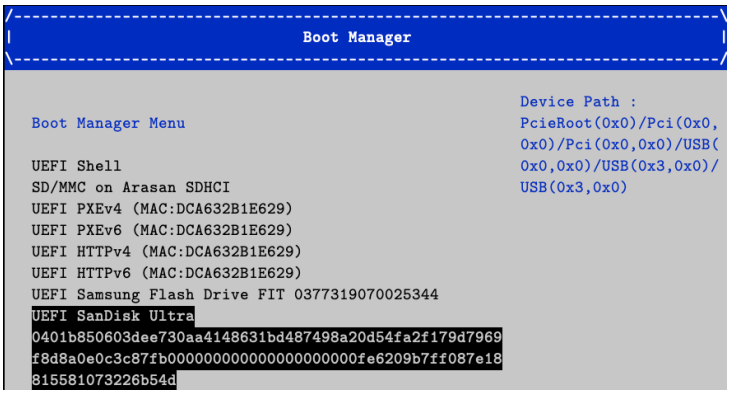
If using a kickstart script, the NIC name for onboard GbE is **vmnic128**.

4.3. Booting the installer

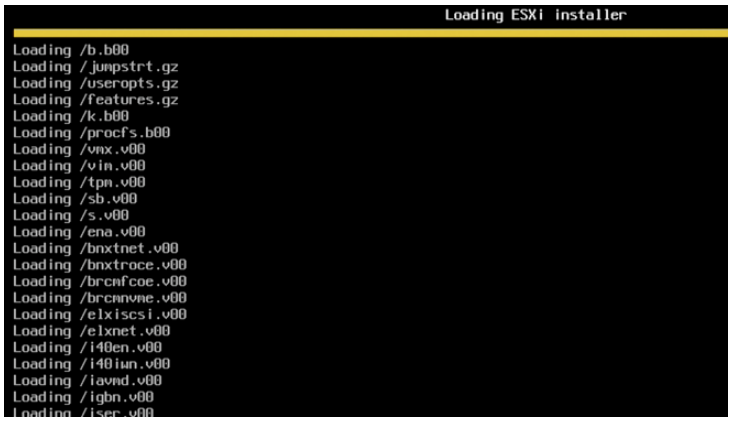
Plug in the USB key with installer into the Pi, and power on (or cycle) the Pi. Enter UEFI configuration by mashing the **ESC** key. Then, use the arrow keys to navigate to **Boot Manager**:



Press **ENTER**, then navigate to the USB drive with the installer.



Press **ENTER**, and the installer will boot:



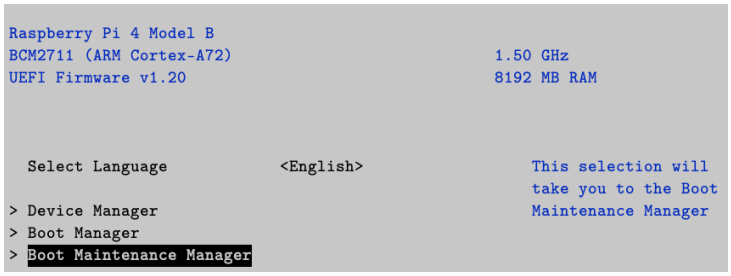
You should be able to follow the generic installation steps.

Note: If you're using the official Raspberry Pi USB keyboard, **F11** is the combination of **Fn** and **F1**.

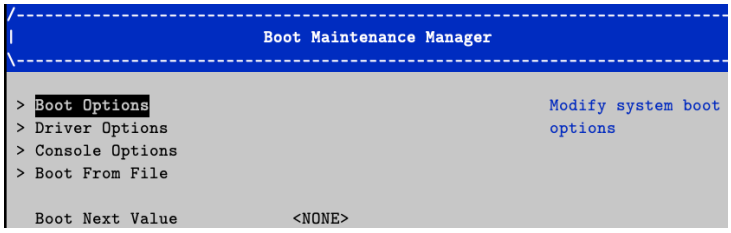
4.4. Post install

The Pi does not have real NVRAM for UEFI boot settings. This means that operating systems like ESXi have read access to the NVRAM, but not write access. The side-effect here is that the ESXi installer will not be able to update boot options, and boot into ESXi may take a really long time as other boot options fail.

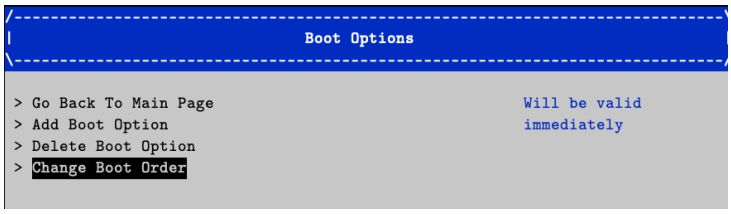
After ESXi install completes, remove the install USB drive. After the system reboots, re-enter UEFI setup and navigate to **Boot Maintenance Manager**:



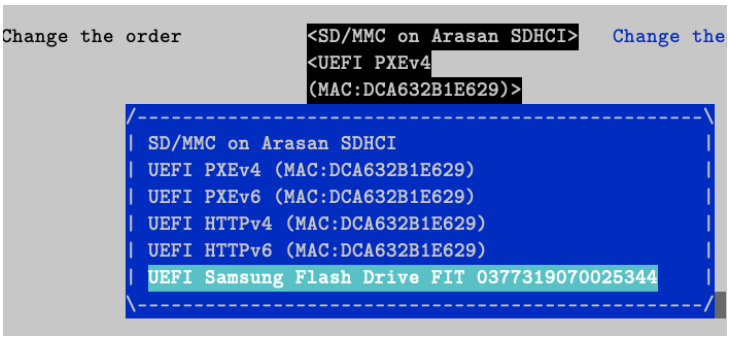
Press **ENTER** and select **Boot Options**:



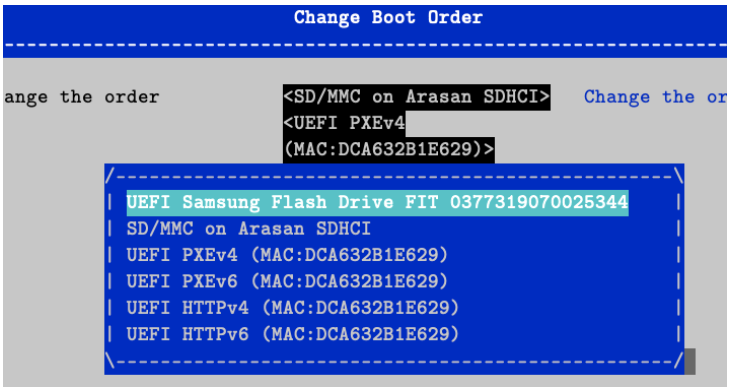
Press **ENTER** and select **Change Boot Order**:



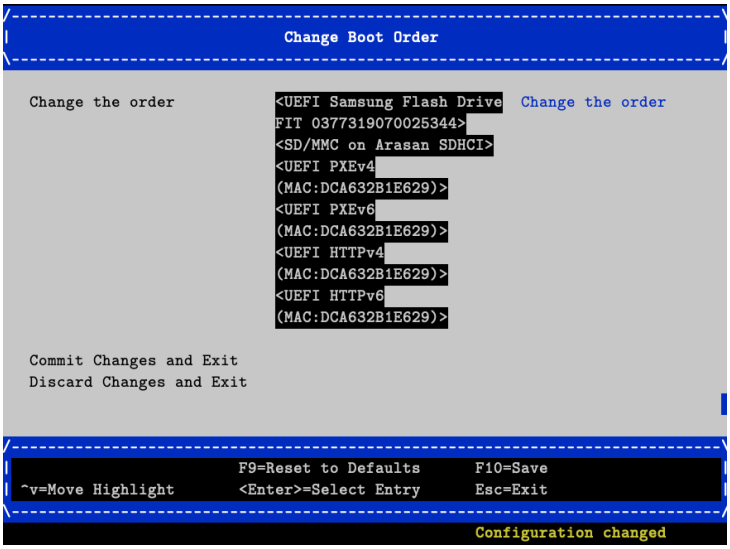
Press **ENTER** twice, and use arrow keys to navigate to the only USB drive you see:



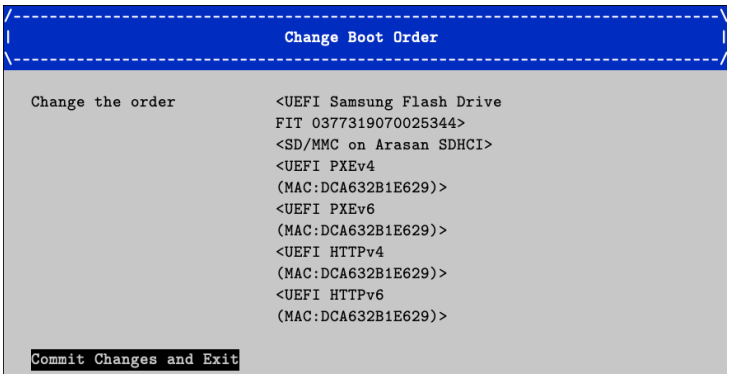
Keep pressing the **+** key until the drive is at the top of the list. You definitely want to skip any network options as they take a long time to time out.



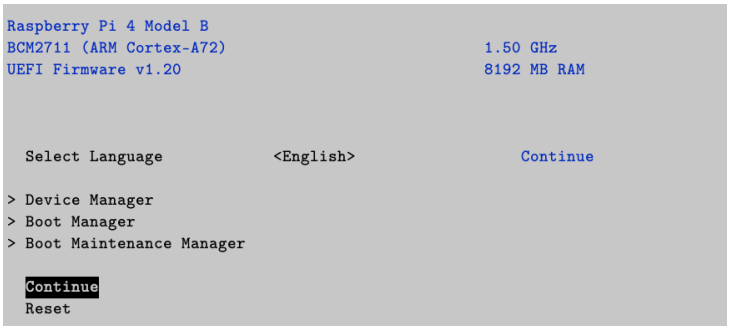
Press **ENTER** to complete the selection.



Now navigate to **Commit Changes and Exit**:



Press **ENTER** and **ESC** out as before to the main UEFI setup screen. Navigate to **Continue**:



ESXi will boot.

4.5. NTP

The Pi does not have a battery backed RTC. Consequently, its notion of time will reset back to the UEFI firmware build date on every boot. Thus, you **must** configure NTP if you wish to add the Pi to a vCenter (ideally, matching the NTP servers used by vCenter to avoid time skew issues).

5. Known issues

5.1. Hardware

5.1.1. Flaky USB in UEFI or ESXi

I/O errors, device not enumerating or disappearing (works in UEFI, not ESXi).

5.1.1.1. This is largely due to power issues.

Workaround: Use a powered hub, especially if using power hungry USB-SATA or USB-NVMe enclosures, or USB NICs with embedded USB hubs.

5.1.1.2. Plugging devices while system is on.

The USB3 implementation on the Pi is a bit sensitive. Could also be due to power fluctuations.

Workaround: Avoid hot plugging devices directly into the Pi if you can.

5.2. UEFI Firmware

5.2.1. Synchronous Exception at 0x00000000371013D8

See https://github.com/pftf/RPi4/issues/97. The **RPI_EFI.FD** image on the SD card, which is the UEFI firmware and the emulated NVRAM, occasionally gets corrupted.

This may be related to EEPROM contents, so be sure to stay up to date (see 3.1).

Workaround: You will have to copy a fresh RPI_EFI.FD over to the SD card.

5.2.2. Official Pi PoE hat fan doesn't work

See https://github.com/pftf/RPi4/issues/101.

Workaround: Either use a different PoE hat or provide active cooling (recommended regardless).

5.3. ESXi-Arm

5.3.1. USB performance

USB3 implementation quirks in the Raspberry Pi 4 and their ESXi workarounds mean significant overhead for some kinds of I/O, such as USB NICs or USB devices pass-through to VMs. USB GbE NICs are known to top out at 200mbps.

Workaround: Use the onboard GbE NIC if possible.