Boot a Linux Disk Image File from GRUB on Ubuntu

Posted **May 27, 2025**
By **Quinoa** **4 min** read

Contents >

Booting a Linux disk image file directly from GRUB can be challenging, especially with GRUB 2's limitations on loading large image files. While GRUB can handle ISO files relatively well, disk image files require a more sophisticated approach. This guide will walk you through creating a bootable Linux disk image and configuring your Ubuntu host system to boot it via GRUB.

Prerequisites

- Ubuntu host system with administrative privileges
- QEMU installed (`sudo apt install qemu-utils`)
- Sufficient disk space for the image file
- A Linux distribution ISO file for installation

Overview

The process involves creating a raw disk image, installing Linux to it, creating a custom initramfs script to mount the image during boot, and configuring GRUB to boot from it.

1. Create a Raw Disk Image

First, create a new raw disk image file using `qemu-img` :

Shell

```
qemu-img create -f raw /path/to/my-linux.raw 80G
```

Replace `/path/to/my-linux.raw` with your desired path and adjust the size as needed.

2. Install Linux to the Image

Create a virtual machine using the raw image as the hard disk:

1. Set up a new VM with qemu
2. Use the raw image file as the primary hard disk
3. Attach your Linux distribution ISO as the CD/DVD drive
4. Boot and install Linux normally to the virtual disk
5. Complete the installation and shut down the VM

3. Mount the Image File on Host System

Now we'll work with the image file from your Ubuntu host system:

Shell

```
# Create a loop device for the image
sudo losetup -Pf /path/to/my-linux.raw

# Check which loop device was assigned
lsblk

# Create mount point and mount the root partition
sudo mkdir -p /mnt/chroot
sudo mount /dev/loop0p2 /mnt/chroot  # Adjust loop device name as needed
```

4. Chroot into the Image

Set up the chroot environment:

Shell

```
# Bind mount essential filesystems
sudo mount --bind /proc /mnt/chroot/proc
sudo mount --bind /sys /mnt/chroot/sys
sudo mount --bind /dev /mnt/chroot/dev
sudo mount --bind /dev/pts /mnt/chroot/dev/pts

# Enter the chroot environment
sudo chroot /mnt/chroot /bin/bash
```

5. Create the Boot Script

Inside the chroot environment, create a custom initramfs script:

</> Shell

```
cd /etc/initramfs-tools/scripts/local-top/
nano mountimg
```

Add the following script content (adjust device paths as needed):

</> Shell

```
#!/bin/sh

PREREQ=""
prereqs() {
    echo "$PREREQ"
}

case "$1" in
    prereqs)
        prereqs
        exit 0
        ;;
esac

# Load NTFS driver if your image is on an NTFS partition
modprobe ntfs3 || modprobe ntfs

# Mount the partition containing your image file
mkdir -p /imgmount
while [ ! -e /dev/nvme0n1p4 ]; do sleep 1; done  # Adjust device path
ntfs-3g /dev/nvme0n1p4 /imgmount || { echo "NTFS mount failed!"; exit 1; }

# Set up loop device for the image
losetup -f /imgmount/my-linux.raw || { echo "Loop setup failed!"; exit 1; }
kpartx -a /dev/loop0

# Mount root partition from the image
mkdir -p /root
mount /dev/mapper/loop0p2 /root || { echo "Root mount failed!"; exit 1; }

# Bind critical virtual filesystems
mount -t proc proc /root/proc
mount -t sysfs sys /root/sys
mount -t devtmpfs dev /root/dev

# Switch to new root
exec switch_root /root /sbin/init
```

Note: If your image file is on an ext4 or other filesystem, modify the mounting commands accordingly:

```shell
# For ext4 filesystems, replace the NTFS lines with:
mount /dev/sda1 /imgmount || { echo "Mount failed!"; exit 1; }
```

Make the script executable:

```shell
chmod +x mountimg
```

6. Update Initramfs

Still in the chroot environment, update the initramfs:

```shell
update-initramfs -u -k all
```

7. Exit Chroot and Copy Kernel Files

Exit the chroot environment:

```shell
exit
```

Copy the kernel and initrd files to your host system:

```shell
# Create directory for extracted files
sudo mkdir -p /boot/extracted

# Copy kernel and initrd files
sudo cp /mnt/chroot/boot/vmlinuz-* /boot/extracted/
sudo cp /mnt/chroot/boot/initrd.img-* /boot/extracted/
```

8. Get the Root Partition UUID

Find the UUID of the root partition within your image:

```shell
sudo blkid /dev/loop0p2 # Adjust loop device name as needed
```

Copy the UUID value for use in the GRUB configuration.

9. Unmount and Release Resources

Clean up the mounted filesystems:

```shell
# Unmount bind mounts
sudo umount /mnt/chroot/proc
sudo umount /mnt/chroot/sys
sudo umount /mnt/chroot/dev/pts
sudo umount /mnt/chroot/dev

# Unmount the main partition
sudo umount /mnt/chroot

# Release the loop device
sudo losetup -d /dev/loop0
```

10. Configure GRUB

Edit the GRUB custom configuration:

```shell
sudo nano /etc/grub.d/40_custom
```

Add a new menuentry (replace with your actual kernel version and UUID):

```shell
```

```
#!/bin/sh
exec tail -n +3 $0

insmod part_gpt
insmod ext2
insmod ntfs

menuentry "Linux (IMG File)" {
    set root=(hd0,gpt2) # Change the actually partition

    linux /boot/extracted/vmlinuz-6.8.0-59-generic root=UUID=your-uuid-here # Ch
    initrd /boot/extracted/initrd.img-6.8.0-59-generic
}
```

Make the file executable and update GRUB:

Shell

```
sudo chmod +x /etc/grub.d/40_custom
sudo update-grub
```

11. Reboot and Test

Reboot your system and select the new menu entry from GRUB to boot into your disk image.

Linux, Boot

Linux boot file grub

Share:

Further Reading

May 25, 2025

Native Boot Windows 11 from a VHDX File

Native booting Windows 11 from a VHDX file enables you to run a full Windows installation directly from a virtual hard disk without virtualization overhead. This approach is particularly useful whe...

OLDER
Increase the Stateful Partition Size of Chrome OS Flex Live USB

NEWER
Using RAG with LLMs for University Quiz Questions