Linux From Debian

Manthan Thakkar

Contents

1 Introduction **3**

1.1 What This Is 3

1.2 Typography 3

2 Preparing the Debian Base **4**

2.1 Creating and Mounting Partitions 4

2.2 Setting Up the Base System 6

3 Piecing Together the Operating System **8**

3.1 Entering `chroot` 8

3.2 Adding Our Packages 11

3.3 Making the System Bootable 12

3.4 Configuring the System 14

4 Wrapping Up **18**

4.1 Deleting the Old OS 18

4.2 User Setup 20

4.3 A Few Final Things 21

Chapter 1

Introduction

1.1 What This Is

This is a walk-through guide in building a custom Debian based Linux distribution.

Basing it upon Debian allows us to skip past the time consuming process of building our desired packages and their dependencies from source (provided we are happy with whatever packages the `apt` package manager offers us), and offers other benefits too (like having a package manager in the first place) which we would not have if building from scratch.

1.2 Typography

Code, commands, names of programs, program and command output, etc. are given in `Computer Modern Typewriter` font.

UI elements such as menu options, button names, etc. are given in *italics*.

Chapter 2

Preparing the Debian Base

2.1 Creating and Mounting Partitions

We start by installing Debian in a VM. We first need to create the root, swap, `/boot`, and `/home` disk partitions for our OS and mount them. Instead of doing this after the Debian installation, it is better to do this during, so that we don't accidentally make breaking changes that force the file system into read-only mode. We create partitions so that OS and user data can be separated. This lets us update the OS without wiping the users' data. It also separates the host from our custom distro.

We will create partitions of 30 GB for the main system and 50 GB for the home partition - not all of the space will be used but we will leave a lot of extra space just in case. The rest will be assigned to our host Debian distribution. Do ensure *Emulated VLAN* is selected as the *Network Mode*. Also make sure your VM has a GPU acceleration supported virtual graphics card.

Upon reaching the *Partition disks* part of the installation, choose:

1. *Guided partitioning*

2. *Guided - use entire disk*
3. Choose the disk you want to use (or the only one there is).
4. *All files in one partition*
5. *#2 [number] GB f ext4*
6. *Delete the partition*
7. Move to where the largest chunk of *FREE SPACE* is and press enter.
8. *Create a new partition*
9. Subtract 80 GB from the number in the text box and press enter.
10. *Beginning*
11. Ensure *Mount point* is set to */*. If not, select it.
12. Done setting up the partition
13. Repeat 7 to 11, but instead of subtracting 80 GB, replace the number with 30 GB and then repeat again but without changing the suggested number.
14. Furthermore, in each repeat use a *Mount point* of *Enter manually* and replace the given path with `/mnt/[distro_name]` on the first repeat and on the third use `/mnt/[distro_name]/home`.
15. Ensure the partitions have been created in such a way that `/` is right after `/mnt/[distro_name]/home` (eg, by first creating the partition `/mnt/[distro_name]` instead of second and doing `/` last). This is because when we eventually get rid of this host OS, its partition needs to be directly after our new OS's `home` partition so that the cleared space can be added to it.

Finish a minimal installation, log in, and then switch user to root.

2.2 Setting Up the Base System

We will now use `debootstrap` to create a minimal Debian system of the latest version. Run the following:

```
apt update
apt install debootstrap

mkdir -p /mnt/[distro_name]/boot/efi
lsblk # Note the NAME mapped to '/boot/efi'
# Usually NAME is 'vda1'
mount /dev/[NAME] /mnt/[distro_name]/boot/efi

debootstrap --verbose --variant=minbase stable \
    /mnt/[distro_name] https://deb.debian.org/debian
```

This will install only the *required* packages including:

- `glibc`
- `gcc`
- `login`
- `shadow`
- `grep`
- `openssl`
- `libpam`
- `apt` and `dpkg`
- `bash`
- `findutils`

- `coreutils` (`ls`, `cp`, `mv`, etc.)
- `util-linux` (`kill`, etc.)
- Archiving tools like `gzip`

and more. However, it does not include *important* packages like `systemd`, networking tools, `vim`, etc. Deleting any of these *required* packages could break the Debian base we are building on top of (so we'll leave them alone). When running `debootstrap`, you could also add `>> ~/log.txt` to the command to save the output and then examine it with `less ~/log.txt` to see what was *required*.

Chapter 3

Piecing Together the Operating System

3.1 Entering `chroot`

We now want to install various packages for our distro. However, we cannot use `apt` straight away, because packages would be installed on the host instead of in the file system of our custom distro. We instead need to use `chroot` to trick the host into thinking that the root of the distro's file system is the root of its own file system, and use the `apt` that `debootstrap` created (instead of the hosts) to install all of our packages.

We also need to mount virtual file systems (these are created by the kernel so that applications in userspace (memory where user programs run) can communicate with it, and exist only in memory) in the new file system so applications in the `chroot` environment can use them (since they can't access the host's copy). To do the mounting, run:

```
mount -v --bind /dev /mnt/[distro_name]/dev
mount -vt devpts devpts /mnt/[distro_name]/dev/pts
mount -vt proc proc /mnt/[distro_name]/proc
mount -vt sysfs sysfs /mnt/[distro_name]/sys
mount -vt tmpfs tmpfs /mnt/[distro_name]/run
# If the directory in the next mount is empty,
```

```
# run the below command.
mount -vt efivarfs efivarfs \
    /mnt/[distro_name]/sys/firmware/efi/efivars
```

Note that:

- `/dev` contains files to make devices (keyboard, disks, etc.) accessible.
- `/devpts` allows access to pseudo-terminals.
- `/proc` provides information from the kernel on processes, memory, etc.
- `/sys` provides information on drivers, kernel modules, busses, etc.
- `/run` stores runtime data such as PID files, sockets, etc.
- `--bind` makes the directory's contents available at the mounted location and avoids duplicating `/dev`'s files.
- `-t` defines the type of file system.
- `-o` defines some permissions. Note that `gid=5` is the *tty* group.

Finally, we can enter the `chroot` environment with:

```
chroot /mnt/[distro_name] /usr/bin/env -i \
    HOME=/root \
    TERM="$TERM" \
    PATH=/usr/local/bin:/usr/bin:/usr/sbin \
    /bin/bash --login
```

Note that:

- `chroot` takes a command as an argument (in our case running `env`).

- `env` takes a command as an argument (in our case starting a `bash` shell).
- `env` clears the environment variables and then adds the ones we specify.
- `HOME` tells programs what the user's home directory is.
- `TERM` tells programs what capabilities the user's terminal has.
- `PS1` sets the default prompt.
- `PATH` tells the shell where to look for executables.
- Double quotes cause the string to be evaluated whilst single quotes don't (the string is taken raw).

You can check if `chroot` was a success by running `ls /mnt` - it should be empty now. If the environment is exited with `exit`, ensure that all the mounted directories are still mounted using `findmnt` before using `chroot`.

Before we end this section, we'll quickly create an `/etc/hosts` file with the following so that `localhost` points to our own machine as well as add Cloudflare's DNS server:

```
printf '127.0.0.1   localhost\n \
        127.0.1.1   [distro_name] \
        ::1         localhost\n' \
    > /etc/hosts
printf 'nameserver 1.1.1.1\n' > /etc/resolv.conf
```

We can also set `root`'s password and add a new user with:

```
passwd
useradd -m -s /bin/zsh -G sudo [username]
passwd [username]
```

3.2 Adding Our Packages

Before continuing, it may be a good idea to clone your VM.

We can now use `apt` to install all the packages we want and build a few from source. For example:

```
apt update
# [arch] in the command below could be arm64, amd64, etc.
apt install --no-install-recommends sudo zsh systemd-sysv \
    systemd-boot parted build-essential man-db manpages \
    procps mlocate firefox-esr vim vifm python3 locales \
    python3-pip foot sway swaylock swayidle xwayland udev \
    waybar sddm pass iwd cups apparmor apparmor-utils git \
    borgbackup wget curl iproute2 wpasupplicant podman \
    uidmap wireguard texlive zathura linux-image-[arch] \
    resolvconf ssh qml-module-qtquick-controls2 \
    qml-module-qtquick-layouts texlive-pictures
```

We then need to register our bootloader with the firmware:

```
cp vmlinuz /boot/efi
cp initrd.img /boot/efi
rm vmlinuz initrd.img vmlinuz.old initrd.img.old
vim /boot/efi/loader/loader.conf
# Change 'default ...-*' to 'default [distro_name].conf'.
rm /boot/efi/loader/entries/[press tab]
rmdir /lost+found

bootctl install
```

3.3 Making the System Bootable

Run `lsblk` and note the *NAME* of the partition where the *MOUNTPOINT* is `/` (eg, `vda5`).

We now need to assign a label to it using `e2label /dev/[NAME] rootfs`. Note that `rootfs` could be anything (provided you also use this below).

Now we create a config file for `systemd-boot` (which we enabled earlier via `bootctl install`) to control the startup and specify the root filesystem. First run `vim /boot/efi/loader/entries/[distro_name].conf` to create the file, and add the following:

```
title   [Distro Name]
linux   /vmlinuz
initrd  /initrd.img
options root=LABEL=rootfs rw quiet
```

To ensure that the partitions are set up correctly for the new OS, use `vim /etc/fstab` and add (after checking correct partitions using `lsblk`, example devices used below):

```
/dev/vda1 /boot/efi vfat umask=0077 0 1
/dev/vda2 / ext4 defaults 0 1
/dev/vda3 swap swap sw 0 0
/dev/vda4 /home ext4 defaults 0 2
```

Now we can:

```
exit
umount -R /mnt/[distro_name]
reboot
```

This should now boot into your distro, and if it doesn't, use `reboot` again and press **Esc** whilst it is trying to boot. A menu should appear where you can choose to go to the *Boot Manager*. Choose the one the contains *systemd* in the name or is called *Linux Boot Manager*.

We can now continue working within our new system rather than using `chroot`, unless the system stops booting properly in which case going back will be needed (select the old boot manager in the menu if required).

The display manager `sddm` should have been started, so logging in by entering a password would start the window manager. Use **CTRL + Enter** to start the terminal emulator. Switch user to `root`.

3.4 Configuring the System

We will first set up networking. Use `ip a` to view existing network interfaces.

To configure a wired network, a `.network` file in `/etc/systemd/network` (eg, `20-wired.network`) and fill it with the following (provided an interface like `eth0` or `enp0s1` exists):

```
[Match]
Name=enp0s1

[Link]
MACAddressPolicy=random

[Network]
DHCP=yes
```

To set up Wi-Fi, first ensure that a wireless interface like `wlan0` exists and run `systemctl enable iwd`, before creating a similar network file to that for wired interfaces, but use something like `wlan0` instead of `enp0s1` and name it something like `20-wireless.network`. Note that files are read in lexical order (so later files matching the same interface won't override earlier ones).

Now use `systemctl enable systemd-networkd` to allow network interfaces to be managed.

Run the following to connect to a wireless network (wired networks should "just work"):

```
iwctl # Launches an interactive tool.
device list # Find available wireless interfaces.
station [device] scan # Scan for available Wi-Fi networks.
# Show the networks discovered by the scan with:
station [device] get-networks
# To connect (may be prompted for password):
```

```
station [device] connect [SSID (network name)]
```

Alternatively, `iwctl station [device] connect [SSID]`. Signal strength can be checked with `iwctl station [device] get-networks`.

Run `vim /etc/default/useradd` and change the default shell to `/bin/zsh`, as well as using `chsh -s /bin/zsh [username]` on all users including root.

We can start editing the appearance of our OS in these files (find the details on the options online):

```
/etc/sway/config
# Add a background image in /etc/sway
/etc/xdg/foot/foot.ini
/etc/xdg/waybar/config
/etc/xdg/waybar/style.css
# Make a theme folder and then a theme in:
/usr/share/sddm/themes/
# And register it in:
/etc/sddm.conf
/etc/zsh/zshrc
# Add colour files to (if wanted):
/usr/share/vim/vim[ver]/colors/
/etc/vifm/colors/

# And set in:
/etc/vim/vimrc
/etc/vifm/vifmrc
/usr/share/vifm/vifmrc
/etc/zathurarc
```

Update locales with:

```
locale-gen C.UTF-8
update-locale LANG=C.UTF-8
```

And then `reboot`.

We can install Firefox extensions globally:

```
# As a regular user
firefox # Install any necessary plugins, eg, uBlock Origin.
# Back as root
updatedb
locate .xpi
# Note all paths that start with '/home/[user]'. For each:
mv [path] /usr/share/mozilla/extensions/[press tab]
```

And then set global Firefox policies:

```
mkdir -p /etc/firefox/policies
vim /etc/firefox/policies/policies.json

# Add the following and save.
{
    "policies": {
        "DisableTelemetry": true,
        "DisableFirefoxStudies": true,
        "DisablePocket": true,
        "PasswordManagerEnabled": false,
        "DNSOverHTTPS": {
            "Enabled": true,
            "ProviderURL": "https://mozilla. \
                cloudflare-dns.com/dns-query"
        }
    }
}
```

Further settings should be customised in `about:preferences`, such as the browser engine and privacy settings.

To configure and use printing, the command line can be used with the `lp`, `lpstat`, `cancel`, and `lpadmin`, or you can use the web interface at `http://localhost:631` and configure printers there in the `Administration` tab's `Add Printer` section. Firefox can be used to execute print commands.

Finally, system information can be done through the following files (ie, branding):

```
/etc/os-release
/etc/issue
/etc/motd
```

Chapter 4

Wrapping Up

4.1 Deleting the Old OS

We have now finished building our Linux distro.

We must now delete the old OS.

```
rm -r /boot/efi/EFI/debian
lsblk # Identify the partition NAME with no MOUNTPOINTS
mkfs.ext4 /dev/[NAME] # eg, vda5
```

Also identify the /home partition and the partitions' sizes in `lsblk`. We should also merge the space used for the old OS with our `/home` partition so that no space is wasted. This is done by deleting the old partition and then adding the new space to the `/home` partition.

```
and then:
   parted # Opens an interactive shell
```

```
print # Note the number corresponding to the
# size we noted down earlier. Eg, 5.
rm 5
resizepart 4 # Enter the size given in brackets.
exit
apt remove --purge parted && apt autoremove

reboot
# Log in
df -h # Should show that the '/home' partition has grown
# to fill the old OS's space.
```

The OS is now ready to be used.

4.2 User Setup

There are a few common things that need to be set up by the user who installs the OS.

First, `firefox`'s `about:preferences` page needs to be visited. Go through the various tabs and enable all the settings that increase security. Also go to the pages of each extension and enable them then run in a private window.

The password manager can be set up using `gpg --full-gen-key`. Noting the public key's ID, use `pass init [ID]` to set up the password store. You can also set up `pass` with `git` for syncing and backups.

For backups, create a desired directory to store backups in, and then run `borg init --encryption=repokey-blake2 [path]`. Make sure to strong permissions on the directory, eg, with `chmod 700 [path]`. Store the password you set in `pass`, or use `pass` to both generate and store the password with `pass generate [path to password file] [length]` and then use that password. It may also be desirable to write a script so `systemd` can automatically take backups as needed.

For VPN services, download a `.conf` file from the VPN provider and then run `wg-quick up [path to .conf file]`. The file must have a valid interface name like `wg0.conf`. Use `wg` to check the VPN's status.

Finally, use `timedatectl set-timezone [Region/City]` to set the timezone, eg, `Europe/London`.

For detail on how to use the tools, refer to their documentation.

4.3 A Few Final Things

Remember to update packages regularly for security reasons, which can be done with `apt update && apt upgrade` or by manually updating packages installed separately.

In case a new Debian version is released, it may be worth rebuilding the OS using the new base, which will likely have newer versions (and therefore more features) for existing packages. Don't forget to take a backup before doing this!