SysAdmin - Part 4 - Linux virtualisation tools & Debian setup

Michel FACERIAS

3 janvier 2024

Polytech Montpellier

Université de Montpellier

Table des matières

1 Linux virtualisation tools **3**
1.1 A few words about virtualisation . . . 3
1.1.1 Concept : Definitions . . . 3
1.1.2 Concept : virtualisation . . . 3
1.1.3 Concept : Paravitualization . . . 3
1.1.4 Concept : Linux has a built-in virtualisation engine . . . 3
1.1.5 Concept : Is there an other virtualisation engine ? . . . 4
1.2 KVM setup . . . 4
1.2.1 To do : Install all dependencies . . . 4
1.2.2 To do : Allow the user to use KVM/QEMU . . . 5
1.2.3 To do : Edit network settings . . . 5
1.2.4 To do : Add a storage pool for your ISO files . . . 5
1.2.5 To do : Create you first VM . . . 6

2 Debian setup **6**
2.1 Why you should use Debian . . . 6
2.1.1 Concept : Debian suites . . . 6
2.1.2 Concept : What are all those names like Etch, Lenny, etc. ? . . . 7
2.1.3 Concept : Where do these codenames come from ? . . . 7
2.2 Debian installation . . . 7
2.2.1 Concept : Network install from a minimal CD . . . 8
2.2.2 Concept : Burning a CD, or making a bootable USB . . . 8
2.2.3 To do : Boot on USB key . . . 8
2.2.4 To do : Chose language . . . 9
2.2.5 To do : Choose the keyboard layout . . . 9
2.2.6 To do : Detect and mount the CD . . . 10
2.2.7 To do : Loading installation components from CD . . . 10
2.2.8 To do : Detecting the network hardware . . . 10
2.2.9 To do : Configure the network . . . 10
2.2.10 To do : Create users and choose passwords . . . 10
2.2.11 To do : Configuring the clock . . . 10
2.2.12 To do : Detecting the disks . . . 11
2.2.13 To do : Partition the disks . . . 11
2.2.14 To do : Install the basic system . . . 12
2.2.15 To do : Configure the package management tool . . . 12
2.2.16 To do : Choose and install software . . . 12
2.2.17 To do : Installation of a boot loader . . . 12
2.2.18 To do : Finish the installation . . . 13
2.3 Post-installation . . . 13
2.3.1 To do : Update the packages source list . . . 13
2.3.2 To do : Software installation . . . 13
2.3.3 To do : Optional : Graphical interface via tasksel . . . 14
2.3.4 Concept : Magical meta-packages . . . 14
2.3.5 To do : Optional - Install *mfa* packages . . . 14
2.3.6 To do : Deep cleaning . . . 15

1 Linux virtualisation tools

1.1 A few words about virtualisation

1.1.1 Concept : Definitions

 When speaking about virtualisation, we separate two entities :

— the **hypervisor** ;

— the **guest**.

The **hypervisor** is the **physical machine** which shares its own resources with the guest(s). The **guest** is the **virtual machine**. It uses the resources shared by the hypervisor to operate an operating system (OS), in the same way as if it was on a physical machine.

1.1.2 Concept : virtualisation

 A **virtual machine** (VM) is, in fact, **a software emulated computer**.

It presents to the guest operating system a **false physical layer** identical to that of a real physical machine. It can **emulate physical hardware** (e.g. Intel e1000 network card) or **hardware that has no real incarnation** (e.g. a specific BIOS or microprocessor).

Emulating **virtual hardware** has a calculation cost for the hypervisor. It is why we try to give the guest a computer architecture as close as possible as the hypervisor's architecture.

1.1.3 Concept : Paravitualization

 **Paravirtualisation** aims at **reducing the cost of harware virtualisation**.

In fact, it will present a **fake hardware** that **only emulates the service to the guest operating system** using directly the hypervisor's system APIs. **This way, there is less load on the hypervisor system**.

We even see, with a *certain other guest system*, **higher performance than** that which would be obtained **on real hardware**.

Of course, it is necessary to have **specific drivers** in the guest system **to use these pseudo-hardwares**.

That's good because, under Linux, these drivers are already **included in the kernel**.

And for a *certain other hosted system*, they have been developed by *RedHat*.

 See more : https://github.com/virtio-win/kvm-guest-drivers-windows

1.1.4 Concept : Linux has a built-in virtualisation engine

 **Kernel-based Virtual Machine** (KVM) is a **full virtualisation solution for Linux** on the x86-64 family hardware containing virtualisation extensions (Intel VT or AMD-V).

It consists of a loadable kernel module, `kvm.ko`, that provides the core virtualisation infrastructure and a processor specific module, `kvm-intel.ko` or `kvm-amd.ko`.

Using KVM, one can run multiple virtual machines running unmodified Linux or a *certain other hosted system* images. Each virtual machine has private virtualized hardware : a network card, disk, graphics adapter, etc.

KVM is open source software. The kernel component of **KVM is included in mainline Linux**, as of 2.6.20.

The userspace companion component of KVM is **QEMU**. QEMU is tasked with handling the virtual machines :

— configurating the parameters ;
— starting and stopping the VM ;
— destroying the VM.

QEMU has a **command-line tool**, but is also available as a **graphical tool**. Both can be used at the same time for the same VM.

 See more : https://www.linux-kvm.org/page/Main_Page

1.1.5 Concept : Is there an other virtualisation engine ?

 As it is explained on its website, *VirtualBox is a powerful x86 and AMD64/Intel64 virtualisation product for enterprise as well as home use... it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 3.*

 See more : https://www.virtualbox.org/

But VirtualBox **is not natively included** in the Linux kernel, So it **taints the kernel**.

 See more : https://www.kernel.org/doc/html/latest/admin-guide/tainted-kernels.html

1.2 KVM setup

There are turnkey solutions to use KVM in production :

— ProxmoxVE is entirely built around KVM/QEMU, and is my prefered one ;
— OpenStack is an other.

But on a **workstation**, it is possible to have a simple and flexible solution, based on KVM/QEMU, but with a **GDI** to to be more user-friendly, while leaving the possibility to access a **command-line interface** and direct acces to the **configuration files**.

1.2.1 To do : Install all dependencies

 As *root* user, use the command line to install needed packages :

```
# apt update && apt install virt-manager qemu-kvm qemu-utils libvirt-daemon-system
    gir1.2-spiceclientgtk-3.0 dnsmasq
```

1.2.2 To do : Allow the user to use KVM/QEMU

Add the user (here `toto`) in these two groups :

```
# adduser toto libvirt
# adduser toto libvirt-qemu
```

The user need to be log out and log in again to join these groups.

1.2.3 To do : Edit network settings

To be in accordance with the *good practices of networking*, use `virsh net-edit default` to change the settings.

```
...
<bridge name='virbr0' stp='off' delay='0'/>
<ip address='192.168.122.254' netmask='255.255.255.0'>
  <dhcp>
    <range start='192.168.122.1' end='192.168.122.199'/>
  </dhcp>
</ip>
...
```

Using `stp=off` allows the disabling of the *spanning tree protocol* and accelerates VM starting.

The services *libvirtd* and *libvirt-guest* need to be reloaded :

```
# service libvirtd restart
# service libvirt-guest
```

1.2.4 To do : Add a storage pool for your ISO files

Historically, ISO files were used to burn a CDROM that was used to start an install process on a physical machine. It's also this file that allows us to create a bootable USB key for the same use.

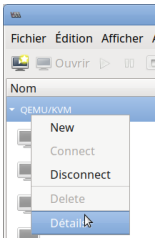
To start the install of a virtual machine, there is no need for a real CDROM or a real USB key. You only need to have the ISO file.

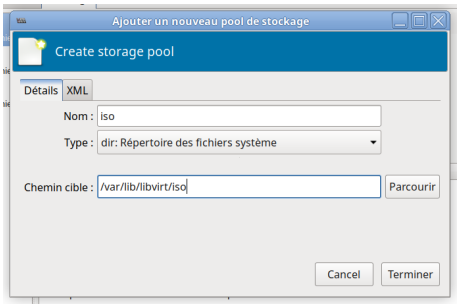
To keep your hypervisor organised, you are going to create a specific storage for the ISO files.

Open *virt-manager* and right-click on the KVM/QEMU domain, then select *Details*.

From the *Storage* tab, click on the + sign on the bottom right-hand corner to **add a new storage pool**.

Use the values detailed below.





Confirm your choices. Then, you will have to **store your ISO files** in `/var/lib/libvirt/iso/`.

1.2.5 To do : Create you first VM

 Using the documentation of **virt-manager**, you are going to install your first VM. You will install a **stable version of Debian**. The installation process is described in the following chapter.

 See more : https://www.tecmint.com/create-virtual-machines-in-kvm-using-virt-manager/

2 Debian setup

According to Wikipedia, **Debian** – also known as **Debian GNU/Linux** – is a Linux distribution composed of free and open-source softwares, developed by the community-supported Debian Project, which was established by Ian Murdock on August 16, 1993.

Debian is **one of the oldest** operating systems **based on the Linux kernel**.

The first version of Debian (0.01) was released on September 15, 1993, and its first stable version (1.1) was released on June 17, 1996.

Debian's current **stable** version is **Debian 12**, and its codename is **Bookworm**.

Debian also forms the basis of many other distributions, notably **Ubuntu, Knoppix, PureOS and Tails** and some professional appliance-based on Linux like **Proxmox VE or Cumulus Network**.

 See more :
https://www.debian.org/intro/why_debian.en
https://www.proxmox.com/en/proxmox-ve
https://en.wikipedia.org/wiki/Cumulus_Networks
https://www.nvidia.com/en-us/networking/ethernet-switching/cumulus-linux/

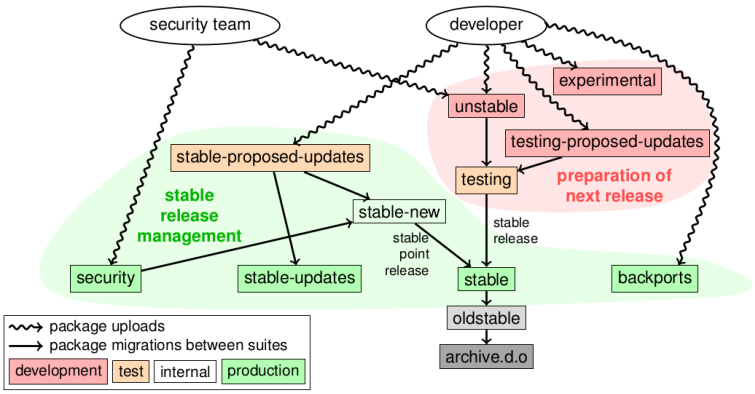
2.1 Why you should use Debian

There are a lot of reasons to choose Debian as your operating system, but mainly :
— Debian is free ;
— Debian is stable and secure ;
— Debian provides smooth upgrades ;
— Debian has a democratic governance structure ;
— ...

 See more : https://www.debian.org/intro/why_debian.en

2.1.1 Concept : Debian suites

 Debian versions are called **suites**. It goes beyond the concept of version.
The flow diagram below summarises how a software goes from a suite to another.



There are three major suites :
— the **stable** suite, the one you should install ;
— the **testing** suite, the future "stable" ;
— the **unstable** suite, which is always in development stage.
Next to these, there is the **oldstable** suite, that's just the **stable** suite's predecessor.
The **experimental** suite is used for packages which are still being developed, and with a high risk of breaking your system. It's used by developers who'd like to study and test bleeding edge softwares.

2.1.2 Concept : What are all those names like Etch, Lenny, etc. ?

 They are just **codenames**. When a Debian suite is in the development stage, it has no version number but a codename. The purpose of these codenames is to make easier the mirroring of the Debian suites.
Currently, **stable** is a symbolic link to **Bookworm** (i.e. Debian GNU/Linux 12) and **testing** is a symbolic link to **Trixie**.
This means that **Bookworm** is the current **stable** distribution and **Trixie** is the current **testing** distribution.
Then, **unstable** is a permanent symbolic link to **Sid**, as **Sid is always the unstable** distribution and **Sid** means *still in development*.

2.1.3 Concept : Where do these codenames come from ?

 So far, they are names of the characters in the **Toy Story** films :
— **Bullseye** (Debian 11 / oldstable) is Andy's wooden toyhorse ;
— **Bookworm** (Debian 12 / stable) is a green toy worm with a built-in flashlight who loves reading books ;
— **Trixie** (Debian 13 / testing) is a blue plastic triceratops ;
— **Sid** is an evil kid known for torturing and destroying his toys.

 See more : https://www.debian.org/doc/manuals/debian-faq/ftparchives.en.html#oldcodenames

2.2 Debian installation

A full installation tutorial is available at **Debian GNU/Linux Installation Guide** :

 See more : https://www.debian.org/releases/stable/amd64/index.en

But we are going to see a **quicker method**.

2.2.1 Concept : Network install from a minimal CD

 A **network install** or **netinstall** CD is a single CD which enables you to install the entire operating system.

This single CD contains just the **minimal amount of software to install the base system** and fetch the remaining packages from the Internet.

The network install assumes that **you have a connection to the Internet**.

But sometimes the **network is not accessible** during the installation, due to lack of firmware. It is **still possible to install an ultra-minimal version** and, after rebooting on it, finish the installation.

The ISO file is at : `https://www.debian.org/CD/netinst/index.en.html#netinst-stable`

2.2.2 Concept : Burning a CD, or making a bootable USB

 **USB memory sticks** have become a commonly used and cheap storage device. Most modern computer systems also allow booting the debian-installer from USB sticks.

Many modern computer systems, especially netbooks and thin laptops, **do not have an optical drive anymore** and booting from a USB stick is the standard way of installing a new operating system on them.

To prepare your USB key on a **Windows** computer, you can use tools like **Rufus**. You can download it at `https://rufus.ie`.

You can also, from a Linux computer, use directly the `dd` tool. Assuming that your USB key is `/dev/sdx`, and `debian.iso` is your netinstall ISO image, use this CLI command :

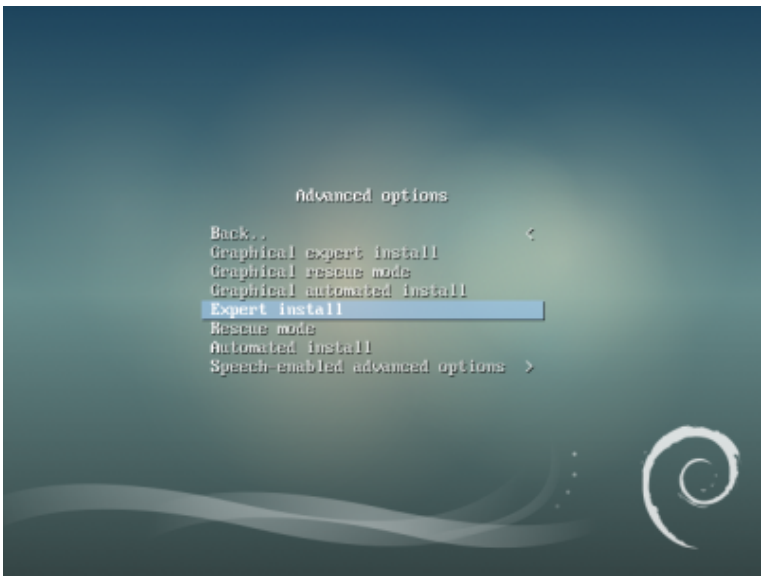
```
# dd if=debian.iso of=/dev/sdx bs=10M status=progress && sync
```

 In the case of a VM install, you only need to add the ISO file to the storage pool created earlier and use it as the install source.

2.2.3 To do : Boot on USB key

 Use the boot setup key. You may have to change the boot order of the PC (F12 on a Dell).

 In the case of a VM install, it will directly boot on the ISO file (if it's correctly configured).



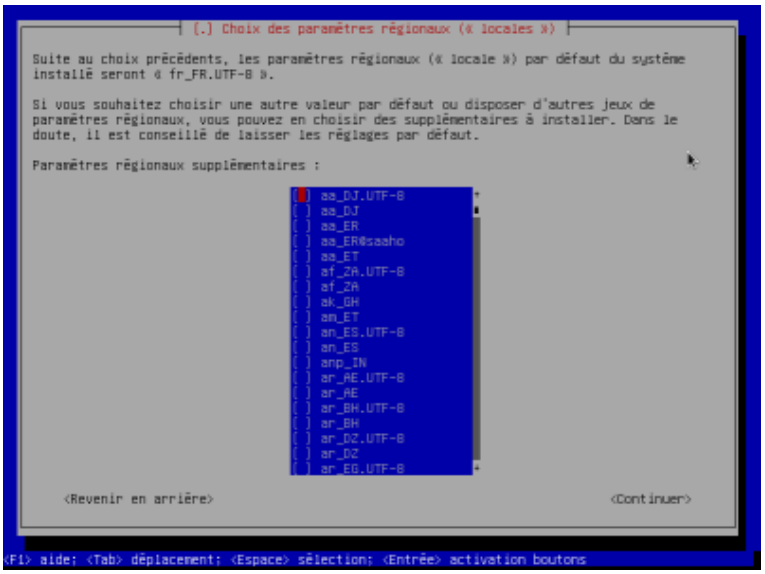
Boot in **expert mode**, by choosing *Advanced options/Expert install* in the boot menu.

2.2.4 To do : Chose language

Assuming you are ok with a French setup :

— Choose *French - Français*
— Choose *France*
— Choose *fr_FR.UTF-8*

Then, on this screen, add nothing and press *enter*.



2.2.5 To do : Choose the keyboard layout

Choose :

— the default keyboard layout (*Français*) ;
— or another one at your convenience.

2.2.6 To do : Detect and mount the CD

 All modules to load are checked by default. You can see :

```
The detected CD: "Debian GNU/Linux 12.1.0 Bookworm - Official AMD64 NETINST ..."
```

2.2.7 To do : Loading installation components from CD

 Usually there is nothing special to install, so please continue.

2.2.8 To do : Detecting the network hardware

 On console #4, by pressing *Alt + F4*, you should see at least *lo* and another network card. Then return to console #1, by pressing *Alt + F1* and continue.

2.2.9 To do : Configure the network

 If you answer **No** to the first page (dhcp), fill in the different fields :

— address,
— mask,
— gateway,
— name servers.

Check and confirm, or reconfigure.

Give a name to the machine (try to avoid **debian**, choose something more specific like **bullseye** for example, or something more personal).

Then, you can leave the domain name empty, especially if you install on a removable USB media.

2.2.10 To do : Create users and choose passwords

 *Do you have to activate the "shadow password" ?* : yes, it's still safer.

Should I allow superuser connections ? : Yes, unless you are a *sudo* aficionado. In which case, install Ubuntu and leave Debian.

Enter *root* for superuser password once, then a second time. Yes, it's not very original, but it can and should be changed afterwards !

Do you have to create an ordinary user account ? : Yes, and I advise you to create this temporary account, which will become UID/GID 1000.

Enter its name (for example *test* user), its login (*test*) and his password (*test*). At least you won't forget it. And it can be changed later too !

2.2.11 To do : Configuring the clock

 Well, the configuration of time in an operating system is out of the scope of this article.

We will consider that your machine is autonomous and that it must set its time as soon as it can access the Internet.

Should I use NTP to set the clock ? : Yes. *ntp server* : leave by default.

Normally, the right time zone is proposed through the language setting. You can adjust if necessary.

2.2.12 To do : Detecting the disks

 This phase is done in silence, so there is nothing to do in particular.

2.2.13 To do : Partition the disks

 I can hear the ayatollahs of */var and /boot separated* screaming in the distance. Good for them !

First, let's make it simple :

— a **root partition** (mounted as /), 30GB is more than enough for a *rootfs* ;

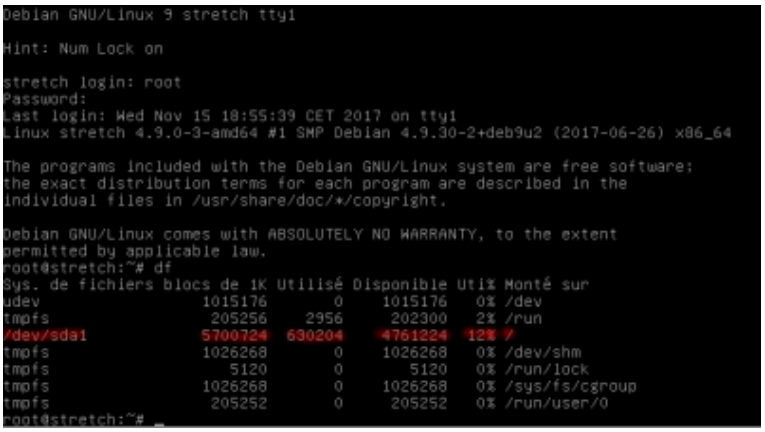
— a **swap**, of the size of the installed RAM.

Nowadays, it's cheaper to buy RAM than to change an SSD disk, so don't overdo it with the swap partition.

 If it is a virtual machine, do not use swap at all. Keep all the disk for the *rootfs*. The installer will claim it, but it is the role of the hypervisor to manage memory.

All data partitions, except swap, will be in *ext4*, this is the standard today.

If you are worried about the size of the *rootfs*, look at the picture below made after the first boot. Yes, a **Debian** is less than 10GB.



Then, we'll do it by hand, so we'll run away from wizards of all kind :

— choose the *manual* mode ;

— search in the list the media you want to work on ;

— partition the media after deleting everything that exists on it (unless of course you have a partition to keep) ;

— on BIOS systems, use primary partitions, whenever possible ;

— on EFI systems, don't forget a small EFI partition (2Gb), at the begining of the disk.

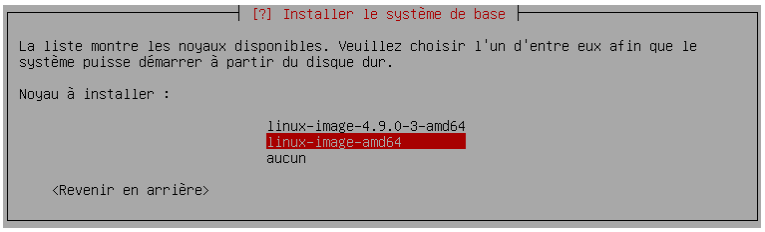
On new media, you will need to make a brand new partition table. You can even use this option to erase the entire configuration of a disk. In this case, **use the menu line that corresponds to the disk** and let yourself be guided.

Choose *ms-dos* as the partition table type for BIOS, unless you choose a *GPT* table for EFI. I know, it looks funny to install Linux with a *ms-dos* partition table, but it's like that on most PCs, even the most recent ones, if you use BIOS boot method.

Once the partitions are created : *Should I apply the changes to the disk ?* : yes

2.2.14 To do : Install the basic system

 At this stage, it is possible to choose between two options :
— a specific kernel (whose version number is given) ;
— a kernel family.



In fact, the **second choice is the right one**. It will allow you to update your kernel, **while the first choice freezes the installed version**.

For the choice of the drivers, it is a question of optimizing the size of the *initrd* file. To gain a few KB, you risk losing a few Ks. At 3.6Ks in an hour, don't waste time on this.

So :
— *Kernel to install* : choose the **generic kernel** (linux-image-amd64) ;
— *Drivers to include* : generic image with all drivers.

2.2.15 To do : Configure the package management tool

 This is when we're going to **deviate from the normal Debian installation** and switch to a special mode.

In less time than it takes a Windows admin to start a server, you will have in your hands a brand new installation, minimal of course, but fully functional !).

Go ahead and follow the instructions :
— *Should we use a mirror on the network ?* : no ;
— Uncheck the *security* and *volatile* packages (by pressing SPACE) ;
— Keep the *backports* unchecked.

2.2.16 To do : Choose and install software

 Well, we didn't add additional installation sources, we have a netinstall CD (150Mb), there shouldn't be much to install.

The main thing is to have the minimum, so : *Software to install* : *Base system.*

2.2.17 To do : Installation of a boot loader

 Choose :
— *Install the GRUB boot loader on a hard disk* ;

— *Install the GRUB boot loader program in the boot sector ?* : Yes ;
— Install GRUB on your first hard drive (usually */dev/sda* on a physical computer).

2.2.18 To do : Finish the installation

 Follow the next steps :
— *Is the system clock in universal time (UTC) ?* : Yes, unless Windows uses the same machine (dual boot), and in that case answer No ;
— then : *Continue*

After ejecting the CD (which must be removed) or removing the USB key that serves as installation media, confirm to restart.

CONGRATS, you have installed your first Debian in expert mode.

2.3 Post-installation

Remember, we did a minimal installation. It goes fast, but there's not much at the end. So, let's finish the job.

2.3.1 To do : Update the packages source list

 Log in as *root*. Modify the `/etc/apt/sources.list` file using the *nano* editor, because at this stage, there is nothing better ! Your file will be like this one :

```
deb http://ftp.debian.org/debian/ bookworm main contrib non-free non-free-firmware
deb http://security.debian.org/debian-security/ bookworm-security main contrib non-free
    non-free-firmware
deb http://ftp.debian.org/debian/ bookworm-updates main contrib non-free  non-free-
    firmware
```

Save with *CTRL+S* and then quit with *CTRL+X*

Update the database of installable packages, then upgrade your distribution :

```
# apt update
  ...
# apt upgrade
  ...
```

The upgrade may ask you to restart some services, accept.

Now you can claim victory. Your Debian is ready to be customized.

2.3.2 To do : Software installation

 You just have to install what you need. This can be done by hand, from a console where you are *root* :
— to search for software : `apt search softwarename`
— to display the details : `apt show softwarename`
— to install : `apt install softwarename`
— to uninstall : `apt remove softwarename`
— to regenerate the database of installable software : `apt update`
— to upgrade all the software of a machine : `apt upgrade`

There is even a **graphical installation manager** *Synaptic*. It will ask you for the *root* password to be able to work. But to use it, you need to install a graphical interface.

2.3.3 To do : Optional : Graphical interface via tasksel

 From the same console, you will launch *tasksel* which will **add a whole bunch of useless packages**.

I don't like it.

It's even contrary to the fundamentals of this tutorial, namely a light and controlled installation.

But, if you want to :

— choose *desktop graphic environment* (and nothing else for the moment) ;

— confirm and let the machine do its job... You can take a long break.

When it is finished, you have to **launch the graphical connection manager** by hand or reboot the machine, using *reboot* command.

2.3.4 Concept : Magical meta-packages

 First, I'm a bit tired of my favorite distribution becoming a *point-and-click game*. I've already left Windows, more than twenty years ago now, because of that, not to find a dummy-friendly system like Ubuntu (CTRL+Z, and my apologies ...).

On an other hand, I note that there is an ecological niche for each animal species. So Ubuntu has its place, and I advise this distribution to my first year students (there is Debian under the hood, after all, so it can't be totally bad).

But I think there is also a place for a friendly, but not bling-bling distribution. So for me, a Debian has to be serious :

— a graphical interface : yes ;

— automations that do what they want behind my back : no.

Then, I like to have a base that can become a server machine easily, and a graphical environment for my daily work. Finally, I'm lazy (like all sysadmins, who are lazy and therefore awesome, unless it's the other way around !).

So, I wanted to automate all this and I created three meta-packages (not so meta) that will make the work more pleasant :

— *mfa-postinstall* ;

— *mfa-desktop* ;

— *mfa-laptop*.

The first one will deliver a ready to use server, with a color prompt, without IPv6, with a clean editor (Vim !)... and other niceties.

The latter will deliver a clean desktop and the other advantages of the first, on which it depends (in the *apt* sense, of course).

So for that, you have to trust my little packages ...

2.3.5 To do : Optional - Install *mfa* packages

You will modify your `/etc/apt/sources.list` to add the fourth line :

```
deb http://ftp.debian.org/debian/ bookworm main contrib non-free non-free-firmware
deb http://ftp.debian.org/debian bookworm-updates main contrib non-free non-free-
    firmware
deb http://security.debian.org/debian-security/ bookworm-security main contrib non-free
    non-free-firmware
deb [trusted=yes] http://hal9000.facerias.org/debian-local/ ./
```

The *[trusted=yes]* allows blind trust in my mirror and its contents !

You will update the list of installable packages using `apt update`, and you can install mfa-postinstall-12, mfa-desktop-12 or mfa-laptop-12 (for Debian 12 / bookworm).

Close and reopen the session, enjoy the color...

That's all folks !

2.3.6 To do : Deep cleaning

After playing with the installation tools, it is a good idea to clean up any residue. Run the next commands :

```
# apt autoremove
  ...
# apt clean
  ...
# dpkg -l | grep ^rc |awk '{print $2}' | xargs dpkg -P
```

The first removes packages that were dependencies of removed packages.

The second cleans the *deb* files, used for installation, but which are no longer mandatory.

The third removes configuration artifacts from removed packages.