

## Comprehensive Guide: Installation of Debian 11 Server on a QEMU Virtual Machine



# Table of contents

- INTRODUCTION.....3**
- A - PREPARATION FOR INSTALLATION.....4**
  - A.1 - Downloading necessary files.....4**
  - A.2 - Writing bash scripts .....5**
- B - DEBIAN 11 INSTALLATION .....9**
  - B.1 - Configuring the installer .....9**
  - B.2 - Moving the disk image.....11**
  - B.3 - Verifying the Debian server.....11**
  - B.4 - Port forwarding and SSH access .....11**
- C - INSTALLATION OF APACHE SERVER, POSTGRESQL, AND PHP.....15**
  - C.1 - Apache Installation.....15**
  - C.2 - PostgreSQL Installation.....16**
  - C.3 - PHP Installation .....22**
  - C.4 - PhpPgAdmin Installation .....23**
- D - SAFETY ANALYSIS.....26**
- E - CONCLUSION .....27**
  - E.1 - Installation summary .....27**
  - E.2 - Final words .....30**

# INTRODUCTION

Welcome to the comprehensive guide on installing a Debian 11 server, without any graphic interface, on a QEMU virtual machine, equipped with functional and accessible Apache, PostgreSQL, and PHP.

In this guide, we will walk you through the step-by-step process of setting up a robust server environment that enables seamless communication between your host machine and the virtual machine. By following these instructions, you will gain the knowledge and confidence to successfully deploy a Debian 11 server with essential components, allowing you to harness the power of Apache, PostgreSQL, and PHP for your web development and data management needs. Let's get started!

# A - PREPARATION FOR INSTALLATION

## A.1 - Downloading necessary files

On a computer with an operating system, start by downloading the Debian 11 ISO images from the following address: <https://cdimage.debian.org/cdimage/release/current/amd64/iso-cd/>

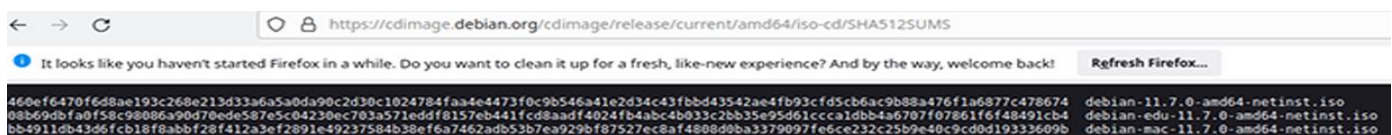
Open a terminal and navigate to the directory where you downloaded your ISO image. Run the command: **sha512sum FILE\_NAME** (refer to Figure A.1.1).

```
lchaal@pc-dg-037-11: /usr/local/images-11$ sha512sum debian-11.7.0-amd64-netinst.iso
4460ef6470f6d8ae193c268e213d33a6a5a0da90c2d30c1024784faa4e4473f0c9b546a41e2d34c43fbb643542ae4fb93cf05cb6ac9b88a476f1a6877c478674  debian-11.7.0-amd64-netinst.iso
lchaal@pc-dg-037-11: /usr/local/images-11$
```

Figure A.1.1

The checksum of your image will be displayed. Go back to the website where you downloaded the image, scroll to the bottom of the page, and open the SHA512SUMS file.

In our case, we are installing the standard version of Debian 11, and the corresponding checksum is found on the first line (refer to Figure A.1.2).



The screenshot shows a web browser window with the address bar displaying <https://cdimage.debian.org/cdimage/release/current/amd64/iso-cd/SHA512SUMS>. Below the address bar, there is a message: "It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!" with a "Refresh Firefox..." button. The main content area displays the SHA512SUMS file, which lists the checksums for various Debian 11 ISO images. The first line is: `4460ef6470f6d8ae193c268e213d33a6a5a0da90c2d30c1024784faa4e4473f0c9b546a41e2d34c43fbb643542ae4fb93cf05cb6ac9b88a476f1a6877c478674 debian-11.7.0-amd64-netinst.iso`. Other lines include `08869dbfa0f58c90806a90d70ede58705c04230ec703a571eddff8157eb441fcd8aadff4024fb4abc4b033c2bb35e95d61ccca1dbb4a6707f07861f6f48491cb4 debian-edu-11.7.0-amd64-netinst.iso` and `bb4911db43d6fcb18f8abbf28f412a3ef2891e49237584b38ef6a7462adb53b7ea929bf87527ec8af4808d0ba3379097fe6ce232c25b9e40c9cd0d19333609b debian-mac-11.7.0-amd64-netinst.iso`.

Figure A.1.2

Compare the two checksums you have and ensure they are identical. If they match, you can proceed to the next step, which involves preparing the Debian installation.

## A.2 - Writing bash scripts

To make it easier to install your virtual machine, we'll start by creating bash scripts to configure our machine and make it easier to install and launch.

First, create a bash script file. Name it "main" and copy the following code into it. Be careful, it will require some modifications on your part:

Lines starting with the '#' symbol are comments, which are not processed by the script. They serve as guidance for modifying your own script, so make sure to read them carefully and take them into account!

### main SCRIPT:

```
#!/bin/bash

# First, we create the image on the local disk.
# After installation, we can move it to a remote server if necessary.

# Enter the location of your downloaded ISO file here
#example: local_image="/datas/Debian-11-.img"

local_image= ""
# Enter the path to the hard drive of your remote server followed by the
name of your ISO
# file here
# exemple: image_nfs="/users/HDD-Storage/images/Debian-11.img"

nfs_image= ""

# There are 4 cases

# If both the NFS image and the local image exist, then the script stops.
if [ -e "$nfs_image" ] && [ -e "$local_image" ]
then
    echo "Unusual situation:"
    echo "You have a local image on this Linux workstation and another on a
remote server."
    echo "Please delete one and start again..."
    exit
fi

# If both the NFS image and the local image do not exist, then a local disk
image of 4GB is # created.

if [ ! -e "$nfs_image" ] && [ ! -e "$local_image" ]
then
    echo "Creating a local disk image on this Linux machine..."
    image="$local_image"
    qemu-img create "$image" 4G
    sync
    echo "Finished."
fi
```

```

# If the NFS image exists, the script displays a message indicating that it
has been found # on the remote server and sets the image variable
accordingly.

if [ -e "$nfs_image" ]
then
    echo "Image found on remote server:"
    image="$nfs_image"
    echo "$image"
fi

# If the local image exists, the script displays a message indicating that
it has been found # on the disk of this Linux machine and sets the image
variable accordingly.
if [ -e "$local_image" ]
then
    echo "Image found on the disk of this Linux station:"
    image="$local_image"
    echo "$image"
fi

# The 'drive' variable is used to define the location and format of the
virtual disk that will
# be used by QEMU to run the virtual machine.

drive="format=raw,file=$image,discard=unmap"

# ISO Image
# the '%' character must be replaced by the location of your ISO file.

iso=$(ls /%/debian-*-netinst.iso)

# QEMU launch command with network installation parameters.
# You can modify these settings to suit your needs.

launch_qemu="qemu-system-x86_64 -M q35 -cpu host -m 4G -enable-kvm -device
VGA,xres=1024,yres=768 -display gtk,zoom-to-fit=off -drive $drive -device
e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::2222-
:22,hostfwd=tcp::4443-:443,hostfwd=tcp::8080-:80,hostfwd=tcp::5432-:5432"

```

Explanation of "launch\_qemu" variable parameters:

'**qemu-system-x86\_64**': Launches the QEMU emulator for the x86\_64 architecture.

'**-M q35**': Specifies the Q35 virtual machine model to use.

'**-cpu host**': Uses the same CPU as the host machine.

'**-m 4G**': Allocates 4GB of RAM to the virtual machine.

'**-enable-kvm**': Activates KVM support for improved performance.

'**-device VGA,xres=1024,yres=768**': Uses a VGA graphics card for the virtual machine with a resolution of 1024x768 pixels.

'**-display gtk,zoom-to-fit=off**': Utilizes the GTK graphical interface to display the virtual machine and disables automatic zooming.

'**-drive \$drive**': Uses the drive variable to specify the hard disk image to be used for the virtual machine.

'**-device e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::2222-:22,hostfwd=tcp::4443-:443,hostfwd=tcp::8080-:80,hostfwd=tcp::5432-:5432**': Adds a virtual network card to the virtual machine and configures the network ports as listed in the table below (refer to Figure A.2.1).

Network service	Virtual machine port	Host station port	Example of use from the host station
SSH	22	2222	\$ ssh login@localhost -p 2222
HTTP	80	8080	URL: http://localhost:8080/
HTTPS	443	4443	URL: https://localhost:4443/
PostgreSQL	5432	5432	\$ psql -h localhost -U postgres postgres

Figure A.2.1

The "main" script we just created will serve as the foundation for developing three additional scripts:

- A script to launch the Debian installation on QEMU.
- A script to launch QEMU.
- A script to move the disk image to a remote server (optional).

If you have correctly written the previous script, there should be no modifications required for these new scripts.

### 1/ move-disk-image:

```
#!/bin/bash

# "main" being the name of the previous script, if you named it
something different, #change it # here

source main

echo "Moving the disk image..."

mv "$local_image" "$nfs_image"

echo "Finished."
```

## 2/ start-installation:

```
#!/bin/bash

# "main" being the name of the previous script, if you named it
something different, #change it # here

source main

# the "-cdrom" option is used to specify the ISO file to be used
for Debian installation.

$launch_gemu -cdrom "$iso"
```

## 3/ launch-virtual-machine:

```
#!/bin/bash

# "main" being the name of the previous script, if you named it
something different,
# change it here

source main

$launch_gemu
```

Now that the scripts created, we can finally proceed with the installation of Debian 11.



## B - DEBIAN 11 INSTALLATION

### B.1 - Configuring the installer

Open a terminal, navigate to the directory where you have created your scripts, and execute the command '**launch-installation**' to run the script with the same name that we just created.

⇒ This will launch the Debian installer within QEMU.

To configure Debian, please follow the steps below. If not specified, leave the options at their default values.

- Language : English
- Location : **other/Europe/France**
- Locales : United States, en\_US.UTF-8
- Keyboard : **French**
- Hostname : The name you want to give to the machine (**refer to Figure B.1.1**).
- Root Password : Enter a strong and unique password. Check the 'Show Password' box to ensure that the entered password is the one you want. This password will be used to log on in superuser mode
- User Account - **Full Name** : Your full name, for example, 'Jean Toto'.
- **User Name** : Enter your login.
- **User Password** : Enter a password to connect to your login. Check the "Show Password" box to make sure the password you enter is the one you want.
- Partition disks : Guided - use entire disk
- Partition disks : All files in one partition
- **Partition disks : Yes**
- **Software Selection** : Make sure that 'Debian desktop' is not checked and 'ssh server' is checked (**refer to Figure B.1.2**).
- Install GRUB : Yes
- **Device for boot loader : /dev/sda**



Figure B.1.1

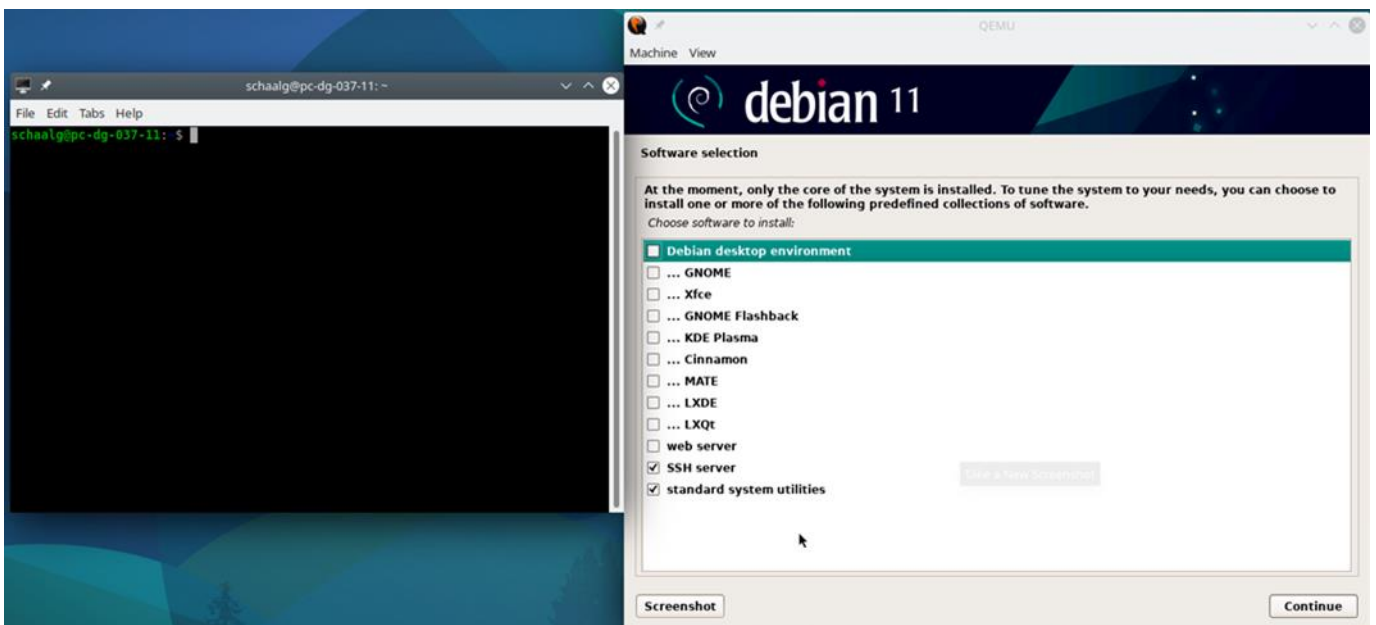


Figure B.1.2

Once the installation is complete, the virtual machine will restart. It is necessary to power it off before proceeding to the next step.

To do this, log on as root and run the following command: **# poweroff**

This is the command you'll need to use to shut down your virtual machine properly after each use.

## B.2 - Moving the disk image

If you want to use this virtual machine from any workstation, it is necessary to move it to a separate server. It is also possible to transfer it to a USB drive, but this is not very practical.

To do this, first make sure that your virtual machine is powered off.

If that's the case, you can then execute the following command: **'move-disk-image'** which you should have created beforehand in **step 2 of part A**.

## B.3 - Verifying the Debian server

You can now restart your virtual machine from a terminal whenever you want by using the command **'launch-virtual-machine'**.

Since we unchecked the 'Debian Desktop' option during the installation phase, the interface of our operating system will be command-line only.

You can verify this by typing the following command: **\$ dpkg -l | grep xorg**

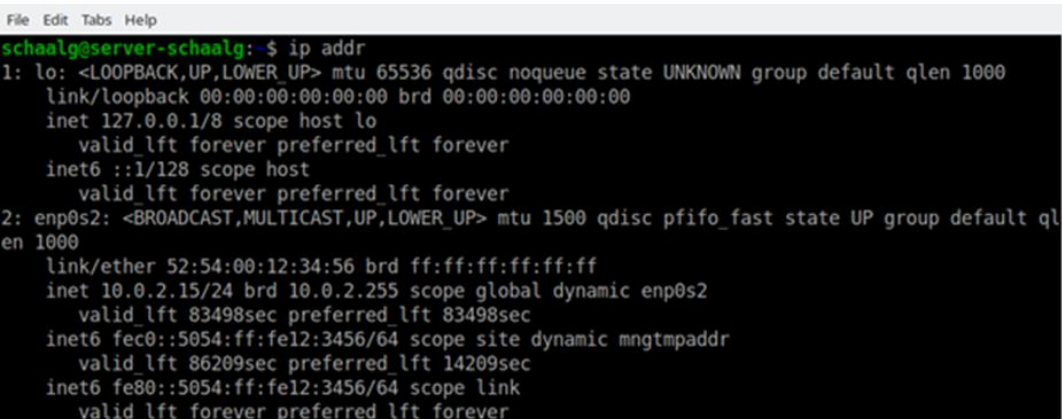
You will notice that the command returns no results, which is completely normal since the xorg packages are only installed if the 'Debian Desktop' option is checked. Xorg being Debian's default display system.

## B.4 - Port forwarding and SSH access

We will now verify that your server is properly configured.

To do this, type the following command: **\$ ip addr**

Locate the presence of your IP address and MAC address as shown in Figure B.4.1.



```
File Edit Tabs Help
schaalg@server-schaalg:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default ql
en 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s2
        valid_lft 83498sec preferred_lft 83498sec
    inet6 fec0::5054:ff:fe12:3456/64 scope site dynamic mngtmpaddr
        valid_lft 86209sec preferred_lft 14209sec
    inet6 fe80::5054:ff:fe12:3456/64 scope link
        valid_lft forever preferred_lft forever
```

*Figure B.4.1*

Now, let's try an external connection by entering the following command: **\$ traceroute www.google.com**

As shown in **Figure B.4.2**, the communication went smoothly

```
File Edit Tabs Help
schaalg@server-schaalg:~$ traceroute www.google.com
traceroute to www.google.com (172.217.20.164), 30 hops max, 60 byte packets
 1 10.0.2.2 (10.0.2.2) 0.065 ms 0.112 ms 0.085 ms
 2 sw-dg-40d-1-tx.iut2.upmf-grenoble.fr (192.168.141.19) 1.426 ms 1.649 ms 1.997 ms
 3 rt-wan.iut2.upmf-grenoble.fr (193.55.51.1) 1.586 ms 2.043 ms 2.227 ms
 4 r-viallet1.grenet.fr (193.54.184.185) 0.906 ms 0.877 ms 0.807 ms
 5 tigr1.grenet.fr (193.54.185.17) 12.219 ms 12.189 ms 12.157 ms
 6 tel-4-grenoble-rtr-021.noc.renater.fr (193.51.181.94) 1.230 ms 1.433 ms 0.902 ms
 7 te0-0-0-1-ren-nr-lyon2-rtr-091.noc.renater.fr (193.51.180.210) 6.861 ms te-0-1-0-12-ren-nr-lyon2-rtr-091.noc.renater.fr (193.51.180.67) 7.260 ms te0-0-0-12-ren-nr-lyon2-rtr-091.noc.renater.fr (193.51.177.57) 6.903 ms
 8 xe-1-0-1-marseille2-rtr-131.noc.renater.fr (193.51.177.196) 13.962 ms reserve-ren-nr-marseille2-rtr-091.noc.renater.fr (193.51.177.23) 7.750 ms reserve-ip8-ren-nr-marseille2-rtr-091.noc.renater.fr (193.51.177.105) 12.846 ms
 9 72.14.218.132 (72.14.218.132) 9.810 ms 9.410 ms 9.318 ms
10 108.170.252.242 (108.170.252.242) 6.925 ms 7.203 ms 74.125.244.216 (74.125.244.216) 6.896 ms
11 216.239.35.209 (216.239.35.209) 14.008 ms 216.239.35.201 (216.239.35.201) 13.695 ms 13.391 ms
12 72.14.238.62 (72.14.238.62) 15.441 ms 209.85.253.216 (209.85.253.216) 15.676 ms 13.335 ms
13 108.170.244.225 (108.170.244.225) 14.108 ms 14.019 ms 14.816 ms
14 142.251.253.33 (142.251.253.33) 12.827 ms 142.251.253.35 (142.251.253.35) 12.928 ms 142.251.253.33 (142.251.253.33) 12.898 ms
15 waw02s07-in-f4.1e100.net (172.217.20.164) 12.854 ms 12.604 ms 12.788 ms
```

**Figure B.4.2**

We have confirmed that communication with the outside world is possible. Let's now try using the SSH protocol and port 2222 provided (see **Figure A.2.1**) to connect to the virtual server from your host machine.

First, let's verify that SSH is running by using the following command (see **Figure B.4.3**):

**# systemctl status ssh**

To enter this command you need to be logged as a super-user. To do this type the following command:

**\$ su -**

In the remainder of this guide, commands will be displayed with a symbol in front, indicating their respective modes:

**\$** User mode

**#** Superuser / root mode

```
root@server-schaalg:~# systemctl status ssh
• ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-05-26 13:09:30 CEST; 9min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 397 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Main PID: 437 (sshd)
    Tasks: 1 (limit: 4661)
   Memory: 3.8M
      CPU: 49ms
   CGroup: /system.slice/ssh.service
           └─437 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

May 26 13:09:29 server-schaalg systemd[1]: Starting OpenBSD Secure Shell server...
May 26 13:09:30 server-schaalg sshd[437]: Server listening on 0.0.0.0 port 22.
May 26 13:09:30 server-schaalg sshd[437]: Server listening on :: port 22.
May 26 13:09:30 server-schaalg systemd[1]: Started OpenBSD Secure Shell server.
```

**Figure B.4.3**

Now that you verified that SSH is active, you can connect to the virtual server from your host machine, by typing the following command: **'ssh login@localhost -p 2222'**

(Be sure to replace 'login' with the login of your Debian server).

You will then be prompted to enter your password (see Figure B.4.4).

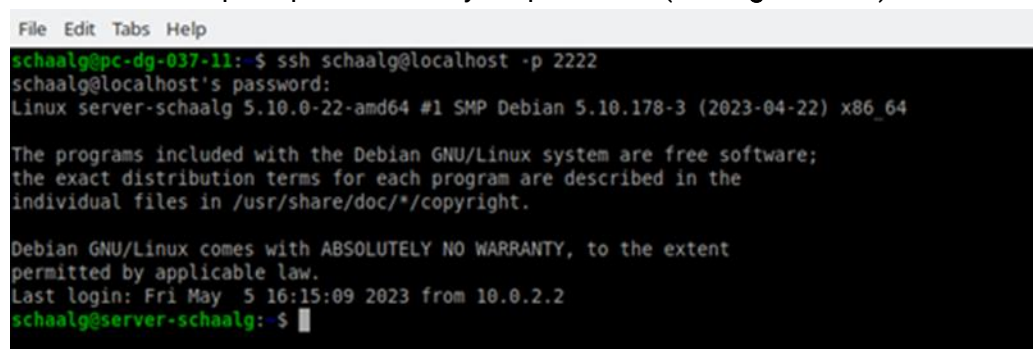
A terminal window with a menu bar (File, Edit, Tabs, Help) and a dark background. The text shows a user 'schaalg' at 'pc-dg-037-11' running 'ssh schaalg@localhost -p 2222'. The prompt changes to 'schaalg@localhost's password:', followed by the system boot information for 'Linux server-schaalg 5.10.0-22-amd64 #1 SMP Debian 5.10.178-3 (2023-04-22) x86\_64'. It then displays the Debian GNU/Linux system's free software notice and warranty disclaimer, followed by the last login time 'Fri May 5 16:15:09 2023 from 10.0.2.2'. The prompt finally changes to 'schaalg@server-schaalg:~\$'.

Figure B.4.4

Now that you are connected from the host machine, we can test if everything is working correctly.

Our objective is to install our first package, which in this case will be the "micro" text editor.

If you haven't already done so, start by switching to superuser mode by typing the following command:  
\$ su -

You will be prompted to enter your superuser password, which you also set during the Debian installation phase. Now you are connected as a superuser!

Let's take advantage of this and install our package by typing the following command:  
# apt install micro

You will be asked to confirm your choice, press the "y" key on your keyboard to proceed (see Figure B.4.5).

Note that this confirmation prompt will appear for the installation of each package in the rest of this guide.

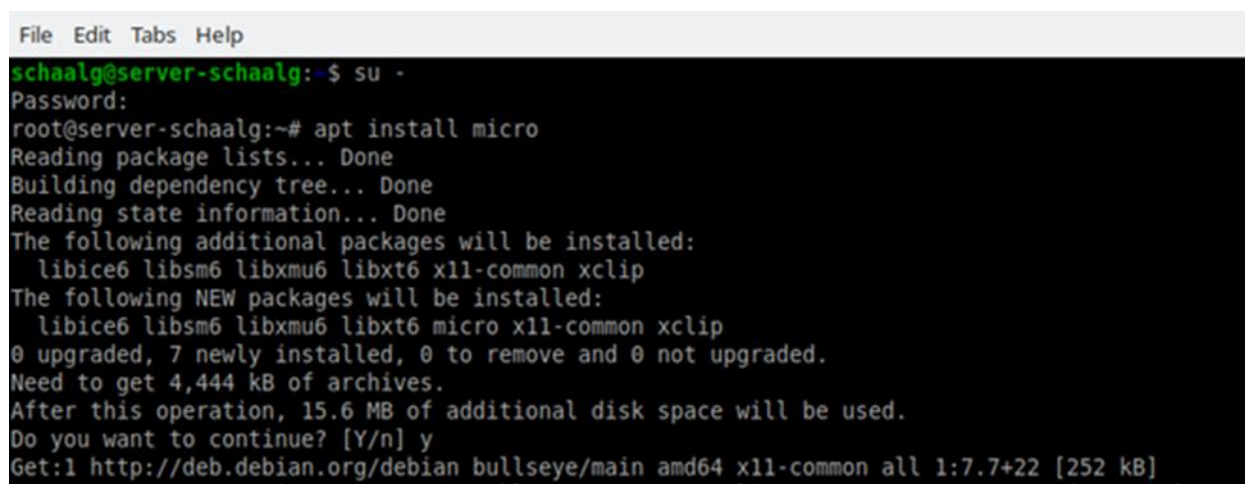
A terminal window with a menu bar (File, Edit, Tabs, Help) and a dark background. The text shows a user 'schaalg' at 'server-schaalg' running 'su -'. The prompt changes to 'root@server-schaalg:~#'. The user then runs 'apt install micro'. The terminal shows the package lists being read, the dependency tree being built, and the state information being read. It then lists additional packages to be installed (libice6, libsm6, libxmu6, libxt6, x11-common, xclip) and the new packages to be installed (libice6, libsm6, libxmu6, libxt6, micro, x11-common, xclip). It shows that 0 packages are upgraded, 7 are newly installed, 0 are to be removed, and 0 are not upgraded. It also shows the disk space requirements: 4,444 kB of archives and 15.6 MB of additional disk space. The prompt asks 'Do you want to continue? [Y/n]' and the user presses 'y'. The terminal then shows the download progress for 'x11-common' from the Debian repository.

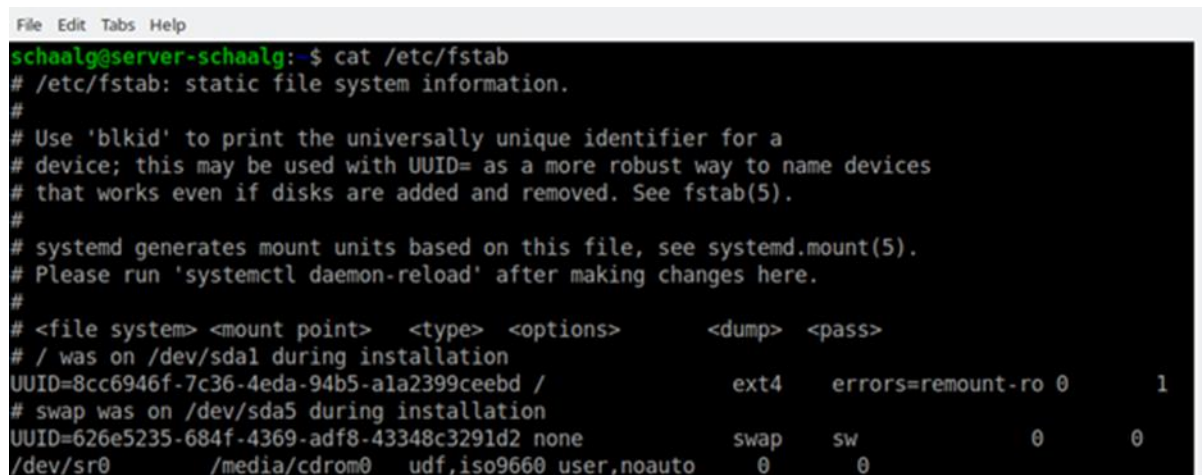
Figure B.4.5

Now let's verify that our package has been installed correctly.

To do this, type the following command: \$ micro /etc/fstab (see Figure B.4.6).

If you have followed the previous steps correctly, the editor should display the contents of the fstab file.

This file is a configuration file used by UNIX-based operating systems, such as Debian, to define the filesystem mounting settings during system startup

A terminal window with a dark background and light text. The window title is "File Edit Tabs Help". The prompt is "schaalg@server-schaalg:~\$". The command "cat /etc/fstab" has been executed, displaying the contents of the file. The output includes comments about static file system information, UUIDs, and a table of filesystem entries.

```
schaalg@server-schaalg:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=8cc6946f-7c36-4eda-94b5-ala2399ceebd / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=626e5235-684f-4369-adf8-43348c3291d2 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

Figure B.4.6



# C - INSTALLATION OF APACHE SERVER, POSTGRESQL, AND PHP

## C.1 - Apache Installation

Apache is a widely used web server software that delivers web content and applications over the internet. It provides stability, flexibility, and performance for hosting websites and supporting various web technologies.

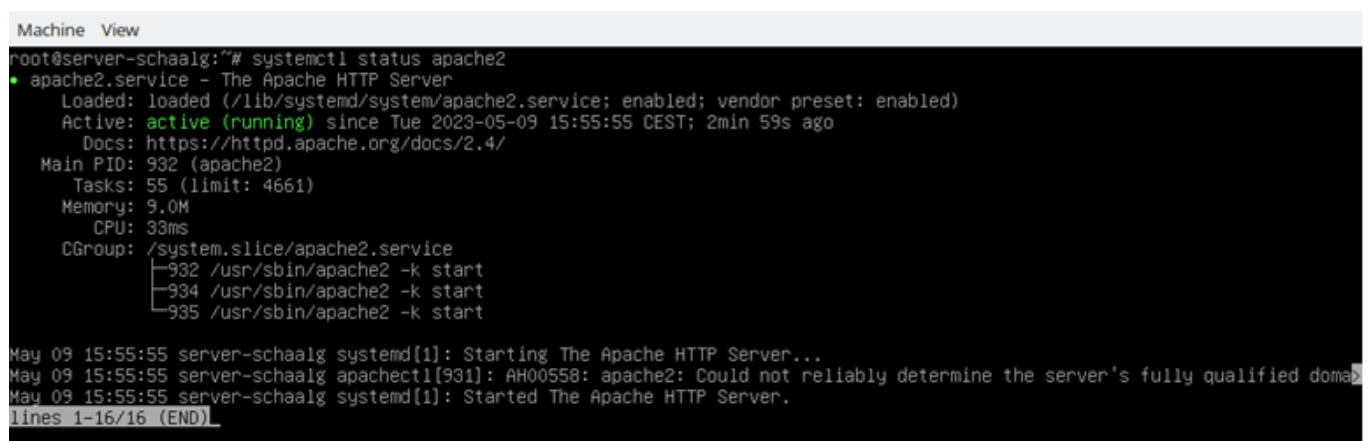
To install Apache on your virtual machine, ensure that you are logged in as a superuser and then execute the following command:

```
# apt install apache2
```

To verify that Apache has been successfully installed, type the following command:

```
# systemctl status apache2 (see Figure C.1.1)
```

If Apache did not start automatically, type the following command: **# systemctl restart apache**

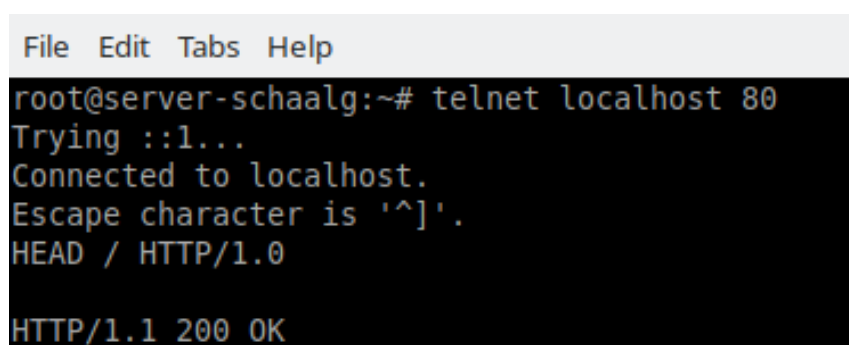
A terminal window titled "Machine View" showing the command "systemctl status apache2" and its output. The output indicates that the apache2.service is loaded and active (running) since Tue 2023-05-09 15:55:55 CEST. It also shows the main PID as 932 and lists tasks for the service. At the bottom, there are log messages from systemd indicating the successful start of the Apache HTTP Server.

```
Machine View
root@server-schaalg:~# systemctl status apache2
• apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-05-09 15:55:55 CEST; 2min 59s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 932 (apache2)
    Tasks: 55 (limit: 4661)
   Memory: 9.0M
      CPU: 33ms
   CGroup: /system.slice/apache2.service
           └─932 /usr/sbin/apache2 -k start
             └─934 /usr/sbin/apache2 -k start
               └─935 /usr/sbin/apache2 -k start

May 09 15:55:55 server-schaalg systemd[1]: Starting The Apache HTTP Server...
May 09 15:55:55 server-schaalg apachectl[931]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, but no IP was set.
May 09 15:55:55 server-schaalg systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

Figure C.1.1

Since your machine is a server without a graphical interface, it is not possible to display an HTML page. However, you can still connect to the Apache server using Telnet by entering the string "HEAD / HTTP/1.0" followed by two line breaks (see Figure C.1.2). The server should respond with "HTTP/1.1 200 OK" if everything is functioning properly.

A terminal window showing a telnet connection to localhost 80. The user enters "HEAD / HTTP/1.0" followed by two line breaks, and the server responds with "HTTP/1.1 200 OK".

```
File Edit Tabs Help
root@server-schaalg:~# telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
```

Figure C.1.2

Although it is not possible to display a web page on the virtual machine itself, it is possible to do so from the host machine.

To achieve this, we will redirect port 8080 from the host machine to port 80, which is the default port for web servers (HTTP protocol, **see Figure A.2.1**).

You can open a web browser on the host machine and access the following URL: **http://localhost:8080**

Verify that you successfully reach the default page of the Apache server on the virtual machine (**see Figure C.1.3**).

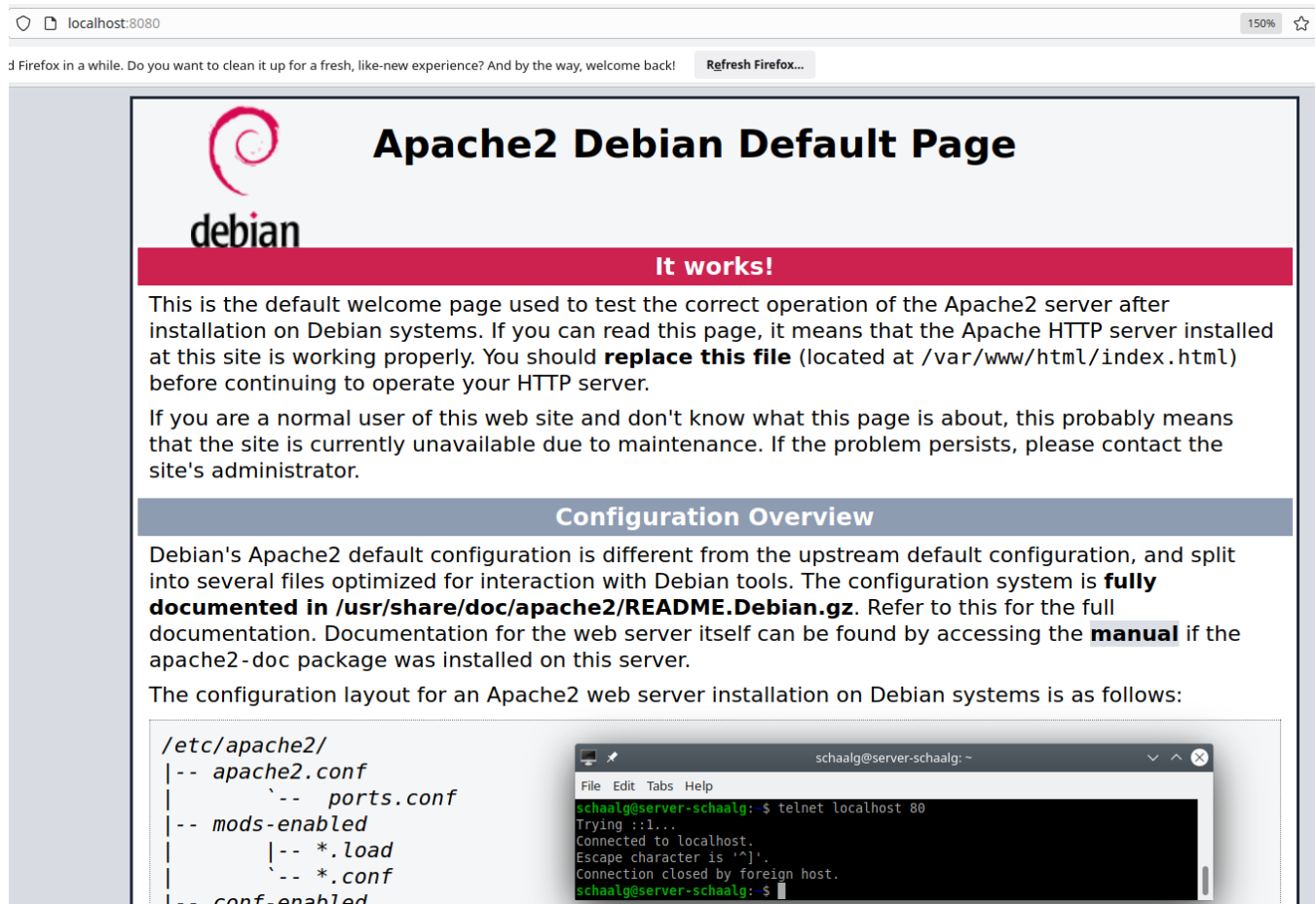


Figure C.1.3

## C.2 - PostgreSQL Installation

PostgreSQL is a powerful open-source relational database management system (RDBMS). It provides a robust and reliable platform for storing, organizing, and retrieving structured data. With advanced features and SQL support, PostgreSQL is widely used for applications that require secure and scalable data storage.

To install PostgreSQL on your virtual machine, ensure that you are logged in as a superuser and then execute the following command: **# apt install postgresql**

You can verify that a postgresQL server program is actually running by typing the following command:  
**\$ pg\_lsclusters**



When the postgresql package was installed, a new unix account was created: postgres. Connect to this account (run from a root shell): **# su - postgres**

Next, we will configure PostgreSQL to be accessible from your Linux workstation. To do this, you need to modify two configuration files and then restart the PostgreSQL server.

Let's start by modifying the "postgresql.conf" file. To do this, execute the following command as a superuser: **# nano /etc/postgresql/13/main/postgresql.conf**

In the Nano text editor, useful commands are displayed at the bottom of the window for quick reference. For example:

- when editing a file in Nano, pressing "Ctrl + O" prompts you to save the changes made to the file.
- additionally, you can use "Ctrl + W" to search for specific text within the file.
- once saved, you can use "Ctrl + X" to exit the editor and return to the command line.

The "postgresql.conf" file configures the PostgreSQL server's configuration parameters. This file is essential for defining server behavior and operating options.

By default, the "**listen\_addresses**" value in the "postgresql.conf" file is usually set to 'localhost' or '127.0.0.1'. This means that the PostgreSQL server only listens to incoming connections from the local IP address (localhost) on the default port, which is 5432.

This restricts connections to the PostgreSQL server from the same system on which it is installed, providing a certain level of security by default. However, this prevents connections from remote systems.

Since we will be connecting to our database from the host machine later on, this line must be modified as follows: **listen\_addresses = '\*'**

By default, the "password\_encryption" parameter in the "postgresql.conf" file is usually set to "md5". This means that user passwords are stored in the PostgreSQL database as MD5 hashes.

However, it is important to note that MD5 hashing is considered relatively weak in terms of security compared to newer and stronger hashing algorithms such as SHA-256. Therefore, we will use SHA-256 instead: **password\_encryption = 'scram-sha-256'**

Please ensure that your "postgresql.conf" file is configured as shown in **Figure C.2.1**.

```

GNU nano 5.4 /etc/postgresql/13/main/postgresql.conf *
# - Connection Settings -
listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5432                     # (change requires restart)
max_connections = 100           # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
# (change requires restart)
#unix_socket_group = ''         # (change requires restart)
#unix_socket_permissions = 0777 # begin with 0 to use octal notation
# (change requires restart)
#bonjour = off                  # advertise server via Bonjour
# (change requires restart)
#bonjour_name = ''              # defaults to the computer name
                                # (change requires restart)

# - TCP settings -
# see "man tcp" for details

#tcp_keepalives_idle = 0        # TCP_KEEPIDL, in seconds;
                                # 0 selects the system default
#tcp_keepalives_interval = 0    # TCP_KEEPIIDL, in seconds;
                                # 0 selects the system default
#tcp_keepalives_count = 0       # TCP_KEEPCNT;
                                # 0 selects the system default
#tcp_user_timeout = 0           # TCP_USER_TIMEOUT, in milliseconds;
                                # 0 selects the system default

# - Authentication -

#authentication_timeout = 1min   # 1s-600s
password_encryption = scram-sha-256 # md5 or scram-sha-256
#db_user_namespace = off

# GSSAPI using Kerberos
#krb_server_keyfile = 'FILE:${sysconfdir}/krb5.keytab'
#krb_caseins_users = off

# - SSL -

```

Figure C.2.1

Now that the server is listening for connection requests from non-local IP addresses, we need to define an authentication rule that will be used for these requests. To do this, edit the authentication rules file:

```
# nano /etc/postgresql/13/main/pg_hba.conf
```

Start by locating the next line: **"#IPv4 local connections:"**

Below that line, add the following rule to only allow authenticated connections with a strong hash function for password storage: **host all all 0.0.0.0/0 scram-sha-256**

Please make sure that your "pg\_hba.conf" file is configured as shown in **Figure C.2.2**:

```

# Database administrative login by Unix domain socket
local all postgres peer

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
host all all 0.0.0.0/0 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256

postgres@server-schaalg:~$ exit

```

Figure C.2.2

To apply these new configurations, restart PostgreSQL using the following command (please note that you need to be logged in as root, not as postgres): **# service postgresql restart**

Let's verify that PostgreSQL is now active by using the following command (refer to **Figure C.2.3**): **# systemctl status postgresql**

```
root@server-schaalg:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Tue 2023-05-09 16:16:20 CEST; 2min 8s ago
     Main PID: 1765 (code=exited, status=0/SUCCESS)
       Tasks: 0 (limit: 4661)
        Memory: 0B
           CPU: 0
      CGroup: /system.slice/postgresql.service

May 09 16:16:20 server-schaalg systemd[1]: Starting PostgreSQL RDBMS...
May 09 16:16:20 server-schaalg systemd[1]: Finished PostgreSQL RDBMS.
root@server-schaalg:~# su - postgres
postgres@server-schaalg:~$ psql
psql (13.10 (Debian 13.10-0+deb11u1))
Type "help" for help.
```

Figure C.2.3

Log back in with the postgres login using the following command from the root account: **# su - postgres**

We're now going to create a postgres user. To do this, type the following command:  
postgres=# **CREATE USER toto;**

Modify the user you've just created, granting all permissions:  
postgres=# **ALTER USER toto WITH superuser createdb createrole password 'password';**

Now let's create a new database with the user we've just created as its owner:  
postgres=# **CREATE DATABASE 'my\_base' WITH OWNER toto;**

Before connecting to your database, check ta you are the owner with the following command (see **Figure C.2.4**):

```
Machine View
postgres=# \l
          List of databases
  Name      | Owner   | Encoding | Collate | Ctype   | Access privileges
-----+-----+-----+-----+-----+-----
ma_base    | schaalg | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres +
           |          |          |          |          | postgres=CTc/postgres
template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres +
           |          |          |          |          | postgres=CTc/postgres
(4 rows)
```

Figure C.2.4

Now connect to the database you've just created with the following command:  
postgres=# **\c my\_base**

Let's create a new table in our database with the following command:

```
my_base=# CREATE TABLE my_table ( id int primary key, name varchar(30) );
```

Let's insert a few rows into our new table with the following command:

```
my_base=# INSERT INTO my_table(id, name)
```

**VALUES**

```
(1, 'toto'),
```

```
(2, 'tata'),
```

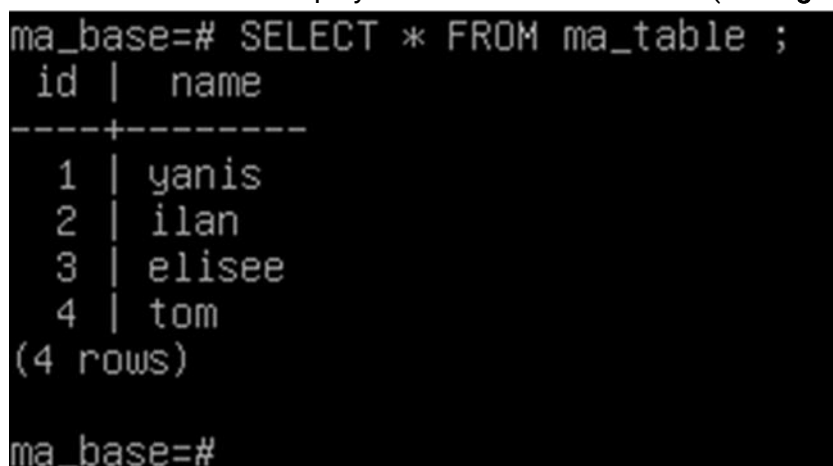
```
(3, 'titi'),
```

```
(4, 'tutu');
```

Finally, let's query our table to check that everything is working with the following command:

```
my_base=# SELECT * FROM my_table
```

The terminal should display a table similar to this one (see Figure C.2.5):



```
ma_base=# SELECT * FROM ma_table ;
 id | name
----+-----
  1 | yanis
  2 | ilan
  3 | elisee
  4 | tom
(4 rows)

ma_base=#
```

*Figure C.2.5*

We're now going to connect to postgresql from the host machine.

To do this, execute the following command: `$ psql -h localhost postgres -U toto`

Here toto is the name of the postgres user we created earlier, replace it with the name of your postgres user.

You will then be asked to enter this user's password. In the case of this guide, toto's password has been defined as 'password'.

We're now going to check that the user password we created earlier has been hashed using the SHA-256 hash function (see Figure C.2.6).

To do this, list the contents of the pg\_shadow system table with the following command:

```
SELECT * FROM pg_shadow;
```

username	usesysid	usecreatedb	usesuper	userepl	usebypassrls	passwd
postgres	10	t	t	t	t	
schaalg	16384	t	t	f	f	SCRAM-SHA-256\$4096:xaE5hmM3ys0zRJMDQRkY6Q==\$1D1FcZIjooRbYSrGpukFHPdTXgMLsPgUPB1Q/HhI6Ho=:0

(2 rows)

Figure C.2.6

You can also connect to the base you previously created and query it from the host machine (see Figure C.2.7)

```
File Edit Tabs Help
schaalg@pc-dg-025-03:~$ psql -h localhost postgres -U schaalg
Password for user schaalg:
psql (13.11 (Debian 13.11-0+deb11u1), server 13.10 (Debian 13.10-0+deb11u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

schaalg@postgres=> \c ma_base
psql (13.11 (Debian 13.11-0+deb11u1), server 13.10 (Debian 13.10-0+deb11u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "ma_base" as user "schaalg".
schaalg@ma_base=> SELECT * FROM ma_table;
 id | name
----+-----
  1 | yanis
  2 | ilan
  3 | elisee
  4 | tom
(4 rows)
```

Figure C.2.7

## C.3 - PHP Installation

PHP is a popular server-side scripting language used for web development. It allows you to create dynamic and interactive web pages by embedding PHP code directly into HTML. PHP can interact with databases, handle forms, generate dynamic content, and perform various other tasks on the server side.

To install PHP on your virtual machine, ensure that you are logged in as a superuser and then execute the following command: **# apt install php-common libapache2-mod-php php-cli**

This command installs PHP and additional packages:

- **php-common**: Provides common files and configurations used by various PHP extensions.
- **libapache2-mod-php**: An Apache module that enables Apache to process PHP files. It integrates PHP with the Apache web server, allowing the execution of PHP scripts on the server.
- **php-cli**: Installs the PHP command-line interface (PHP CLI), which allows executing PHP scripts directly from the command line without relying on a web browser.

By installing these packages, you ensure that PHP is properly set up on your system for web server integration and command-line script execution.

To test your PHP installation, follow these steps:

1. Ensure that you are logged in as a superuser and create a file named `info.php` in the `/var/www/html/` directory by using the following command:  
**# nano /var/www/html/info.php**
2. The nano editor will then open the blank php file you've just created. Place the following code inside it:  

```
<?php
phpinfo();
phpinfo(INFO_MODULES);
?>
```
3. Access the URL **`http://localhost:8080/info.php`** from the host machine.

This opens a page containing the main features of your PHP installation (see **Figure C.3.1**)


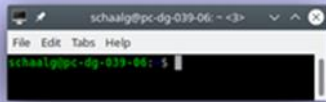
localhost:8080/info.php		150%
PHP Version 7.4.33		
System	Linux server-schaalg 5.10.0-22-amd64 #1 SMP Debian 5.10.178-3 (2023-04-22) x86_64	
Build Date	Feb 22 2023 20:07:47	
Server API	Apache 2.0 Handler	
Virtual Directory Support	disabled	
Configuration File (php.ini) Path	/etc/php/7.4/apache2	
Loaded Configuration File	/etc/php/7.4/apache2/php.ini	
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d	
Additional .ini files parsed	 /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini	
PHP API	20190902	
PHP Extension	20190902	
Zend Extension	320190902	
Zend Extension Build	API320190902,NTS	
PHP Extension Build	API20190902,NTS	
Debug Build	no	
Thread Safety	disabled	
Zend Signal Handling	enabled	
Zend Memory Manager	enabled	

Figure C.3.1

In this setup, we use the `/var/www/html/` directory as the location for serving web content. This directory is the default root directory for the Apache web server on Debian-based systems. By placing the `info.php` file in this directory, we make it accessible to the web server.

The port number 8080 is used because we previously redirected port 8080 of the host machine to port 80 of the virtual machine. Port 80 is the default port for HTTP communication, and by redirecting it, we can access the web server running inside the virtual machine from the host machine.

The term "localhost" refers to the loopback network interface of the host machine. When accessing `http://localhost:8080` from the host, it is resolved to the virtual machine's IP address, allowing us to connect to the web server running on the virtual machine.

## C.4 - PhpPgAdmin Installation

phpPgAdmin is a web-based administration tool for managing PostgreSQL databases. Built using PHP and designed to work with Apache, phpPgAdmin allows users to interact with PostgreSQL databases through a user-friendly web interface. It leverages the capabilities of PHP to provide a convenient way to perform common database management tasks such as creating, modifying, and deleting database objects, executing SQL queries, and managing user privileges.

To install phpPgAdmin on your virtual machine, ensure that you are logged in as a superuser and then execute the following command: `# apt install phppgadmin`



Let's start by modifying the "config.inc.php" file. To do this, type the following command while logged in as root: # **nano /etc/phppgadmin/config.inc.php**

The "/etc/phppgadmin/config.inc.php" file is a crucial configuration file for the PhpPgAdmin web application. It is responsible for securely defining the connection parameters to the PostgreSQL database. This file plays a critical role in ensuring the proper functioning of PhpPgAdmin by establishing a direct link between the web interface, PostgreSQL, and Apache.

We will need to modify the following lines:

```
// Display name for the server on the login screen

$conf['servers'][0]['desc'] = 'PostgreSQL 13';


// Hostname or IP address for server. Use '' for UNIX domain socket.

// use 'localhost' for TCP/IP connection on this computer

$conf['servers'][0]['host'] = 'localhost';


// Database port on server (5432 is the PostgreSQL default)

$conf['servers'][0]['port'] = 5432;
```

Set extra\_login\_security value to false:

```
$conf['extra_login_security'] = false;
```

Now let's modify the "phppgadmin.conf" file. To do so, type the following command while logged in as root: # **nano /etc/apache2/conf-enabled/phppgadmin.conf**

The "phppgadmin.conf" file is used to configure the integration of PhpPgAdmin with the Apache web server. It allows you to specify aliases, paths, and access permissions required for running PhpPgAdmin through Apache. This configuration file ensures that Apache can properly serve PhpPgAdmin and handle the necessary access and routing for the application.

We will need to modify the following lines:



```

</IfModule>

AllowOverride None

# Only allow connections from localhost:

Require all granted

<IfModule mod_php.c>

```

Now that these two files have been modified, restart the apache webserver using the following command as root: **# systemctl restart apache2**

Once successfully installed, open your web browser from the host machine and access the PhpPgAdmin web console using the following URL: **http://localhost:8080/phppgadmin**

Congratulations, you're now on the PhpPgAdmin home page.

Please enter the username and password of the PostgreSQL user that you created earlier. In the case of our guide we used the login 'toto' and the password 'password'.

You will then be connected to your database from the PhpPgAdmin web interface. From this interface, it is possible to manipulate and modify our database as we did previously on the command line.

For example, let's try the same query we made on the command line:

```
SELECT * FROM my_base;
```

As you can see, the result displayed is identical to the previous one (see Figure C.4.1).

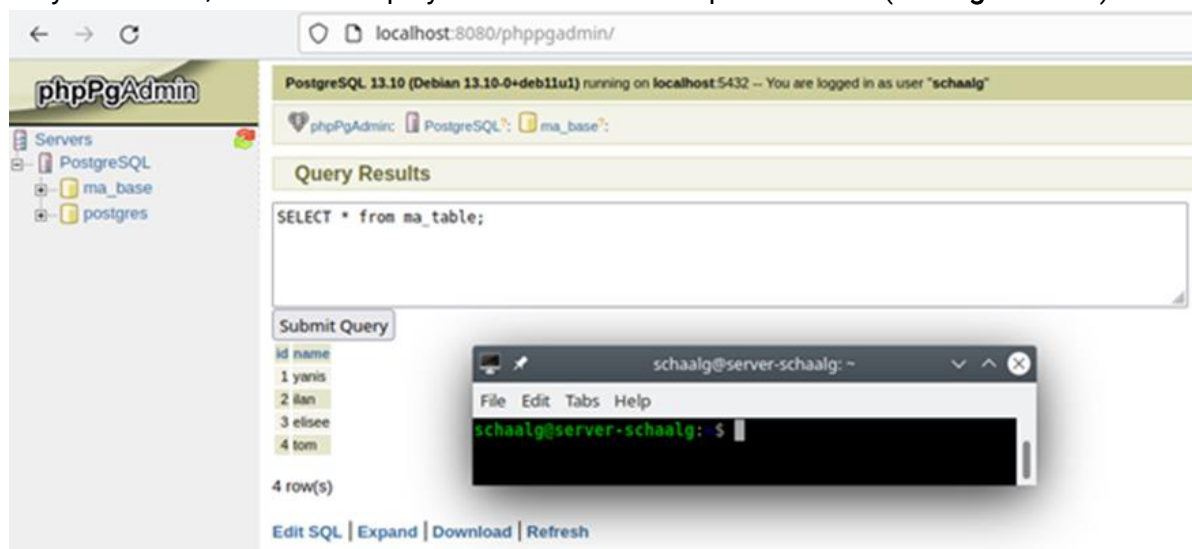


Figure C.4.1

## D - SAFETY ANALYSIS

In order to ensure the security of your Debian server and the installed software packages, it is important to implement the following measures:

1. Enforce strong password policies for user accounts: Set up password complexity requirements and encourage users to use unique, secure passwords to protect their accounts.
2. Configure regular backups of your data: Implement a reliable backup strategy to safeguard your important data in case of any unforeseen incidents or data loss.
3. Implement a firewall: Set up a firewall to restrict unauthorized access to your server. Configure rules to allow only necessary network traffic and block potential threats.
4. Keep your system up to date: Regularly install security updates for your Debian system. Utilize package management tools like apt or apt-get to apply available security updates. Make sure to keep your Debian system updated by installing the latest updates on a regular basis.
5. Perform regular package updates: Keep the installed packages, such as Apache, PHP, and PostgreSQL, up to date by regularly applying updates provided by the package maintainers. This ensures that any known security vulnerabilities or bugs are addressed promptly.

By implementing these measures, you can enhance the security of your Debian server and minimize the risks associated with software vulnerabilities. Regularly updating your system and installed packages will help to maintain a secure and stable server environment.

Remember to stay vigilant and stay informed about security best practices. Stay connected with the Debian community and security resources to stay updated on the latest security advisories and recommendations.

By following these guidelines, you can help ensure the safety and security of your Debian server and the sensitive data it hosts.

# E - CONCLUSION

## E.1 - Installation summary

To conclude this guide, we will create a PHP file that will display all the installations we have performed since the beginning of this guide.

Insert the following php code in this file:

```
<html>
<head>
<title>My installations</title>
</head>
<body>
<?php echo '<p>Hellor</p>'; ?>
<p>
I am
<?php passthru("whoami"); ?>
</p>
<p>
Who is connected ?
<pre>
<?php passthru("who"); ?>
</pre>
</p>
<p>
My disks are
<pre>
<?php passthru("/sbin/blkid"); ?>
</pre>
</p>
<p>
My interfaces
<pre>
<?php passthru("ip addr"); ?>
</pre>
</p>
<p>
<?php passthru("ip addr"); ?>
</pre>
</p>
<p>
My apache install is
<pre>
<?php passthru("dpkg -l | grep apache"); ?>
</pre>
</p>
<p>
My apache status is
<pre>
<?php passthru("systemctl status apache2"); ?>
</pre>
</p>
<p>
```

```

My postgresql status is
<pre>
<?php passthru("systemctl status postgresql"); ?>
</pre>
</p>
<p>
My ssh install is
<pre>
<?php passthru("dpkg -l | grep ssh"); ?>
</pre>
</p>
<p>
My ssh status is
<pre>
<?php passthru("systemctl status ssh"); ?>
</pre>
</p>
</body>
</html>

```

Make sure you have saved your changes, and then go to the command line as a superuser and execute the following command: **# /sbin/blkid**

Open your web browser from the host machine and access the php file we just created using the following URL: **[http://localhost:8080/my\\_installations.php](http://localhost:8080/my_installations.php)**

And there you have it! The web page displayed summarizes all the packages we have installed and configured in this guide!



```

<pre>
Bonjour

Je suis www-data

Qui est connecté ?

schaalg tty1 Jun 1 10:05
schaalg pts/0 Jun 1 10:21 (10.0.2.2)

Mes disques sont

/dev/sda1: UUID="8cc6946f-7c36-4eda-94b5-a1a2399ceebd" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="c2d138f4-01"
/dev/sda5: UUID="626e5235-684f-4369-adf8-43348c3291d2" TYPE="swap" PARTUUID="c2d138f4-05"

Mes interfaces

1: lo: mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s2: mtu 1500 qdisc pfifo fast state UP group default qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s2
        valid_lft 84109sec preferred_lft 84109sec
    inet6 fec0::5054:ff:fe12:3456/64 scope site dynamic mngtmpaddr
        valid_lft 86344sec preferred_lft 14344sec
    inet6 fe80::5054:ff:fe12:3456/64 scope link
        valid_lft forever preferred_lft forever

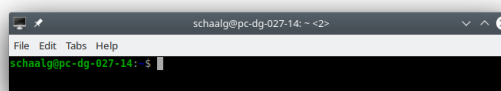
```

My apache install is

ii	apache2	2.4.56-1-deb11u2	amd64	Apache HTTP Server
ii	apache2-bin	2.4.56-1-deb11u2	amd64	Apache HTTP Server (modules and other binary files)
ii	apache2-data	2.4.56-1-deb11u2	all	Apache HTTP Server (common files)
ii	apache2-utils	2.4.56-1-deb11u2	amd64	Apache HTTP Server (utility programs for web servers)
ii	libapache2-mod-php	2:7.4+76	all	server-side, HTML-embedded scripting language (Apache 2 module)
ii	libapache2-mod-php7.4	7.4.33-1+deb11u3	amd64	server-side, HTML-embedded scripting language (Apache 2 module)

My apache status is

```
* apache2.service - The Apache HTTP Server
Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
Active: active (running) since Thu 2023-06-01 10:04:29 CEST; 38min ago
Docs: https://httpd.apache.org/docs/2.4/
Process: 421 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
Main PID: 463 (apache2)
Tasks: 9 (limit: 4661)
Memory: 26.1M
CPU: 439ms
CGroup: /system.slice/apache2.service
├─463 /usr/sbin/apache2 -k start
├─481 /usr/sbin/apache2 -k start
├─482 /usr/sbin/apache2 -k start
├─483 /usr/sbin/apache2 -k start
├─484 /usr/sbin/apache2 -k start
├─485 /usr/sbin/apache2 -k start
├─706 /usr/sbin/apache2 -k start
├─905 sh -c systemctl status apache2
└─906 systemctl status apache2
```



My postgresql status is

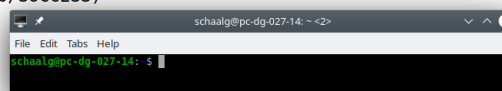
```
* postgresql.service - PostgreSQL RDBMS
Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
Active: active (exited) since Thu 2023-06-01 10:04:32 CEST; 38min ago
Process: 535 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
Main PID: 535 (code=exited, status=0/SUCCESS)
CPU: 885us
```

My ssh install is

ii	libssh2-1:amd64	1.9.0-2	amd64	SSH2 client-side library
ii	openssh-client	1:8.4p1-5+deb11u1	amd64	secure shell (SSH) client, for secure access to remote machines
ii	openssh-server	1:8.4p1-5+deb11u1	amd64	secure shell (SSH) server, for secure access from remote machines
ii	openssh-sftp-server	1:8.4p1-5+deb11u1	amd64	secure shell (SSH) sftp server module, for SFTP access from remote machines
ii	task-ssh-server	3.68+deb11u1	all	SSH server

My ssh status is

```
* ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
Active: active (running) since Thu 2023-06-01 10:04:29 CEST; 38min ago
Docs: man:sshd(8)
     man:sshd_config(5)
Process: 428 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 458 (sshd)
Tasks: 1 (limit: 4661)
Memory: 5.4M
CPU: 84ms
CGroup: /system.slice/ssh.service
└─458 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
```



You can check how much storage space you have left on your virtual machine with the following command (see Figure E.1.1): `$ df -h`

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	1.9G	0	1.9G	0%	/dev
tmpfs	392M	492K	392M	1%	/run
/dev/sda1	3.0G	1.4G	1.4G	50%	/
tmpfs	2.0G	16K	2.0G	1%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	392M	0	392M	0%	/run/user/1000

Figure E.1.1

## E.2 - Final words

In this comprehensive guide, we have covered the installation and configuration of a Debian server with Apache, PHP, and PostgreSQL.

Throughout this guide, we have learned how to install the necessary software, perform basic configurations, and enhance the security of our installation. By following this guide, you have acquired the skills to establish a robust and secure web server environment.

You have learned how to configure Apache for web hosting, install and configure PostgreSQL for database management, and integrate PHP for dynamic web application development.

Moreover, we have addressed important aspects of security, such as configuring security parameters, securing remote access, managing users, and applying security updates.

We hope that this guide has been helpful to you and has provided you with the necessary knowledge to successfully start your own web server. Remember to keep your installation up to date by regularly applying security patches and following recommended best practices.

We appreciate your commitment to following this guide, and we encourage you to continue exploring and deepening your knowledge in web server management and computer security. If you have any further questions, don't hesitate to seek assistance from the community or consult available online resources.

Wishing you continued success in your journey of server management and computer security!

Thank you for following this guide.