SAE S2.03 : Debian 12 server installation guide with Apache, PostgreSQL and PHP

GENTIL Quentin
BUT1 C2

Table des matières

I. Introduction

In this document, we will tackle the installation of a Debian 12 server along with some packages you might already be familiar with. Those include :

- Apache2, which is used for HTTP web servers ;
- PostgreSQL, which you've used during R1.05 as a user ;
- the PHP module for Apache2, which will allow the web servers to render PHP pages ;
- PhpPgAdmin, which helps setting up databases in PostgreSQL through a web application.

Even if it seems complicated, you will be guided through each step of the process, and will be using your knowledge from the R1.04 module. In only a half-hour, everything will be set up!

II. Installing Debian 12

II.1. Setting up

First, you'll need a Debian 12 disk image and a virtual machine (you will be using QEMU/KVM).

Let's start by downloading everything needed, and let's start with the usual :

- # apt update
- # apt upgrade
- # apt clean

Remember to keep your system packages up to date !

In case your system doesn't have QEMU installed :

- # apt-get install qemu-system

Then the installation image can be found **here**.
At the bottom, you may download the « debian-12.10.0-amd64-netinst.iso » file (633MB)

You may want to check the SHA512SUMS file, which contains the SHA512 checksums for the installation images provided. Compare the first one with the checksum you get with the following command :

- $ sha512sum yourInstallationImage.iso

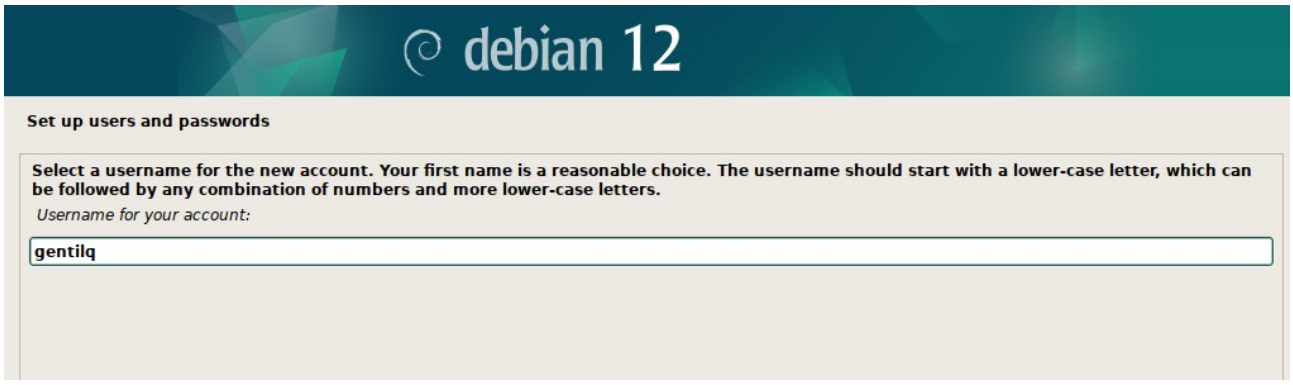
If they do not match, please make sure you have downloaded the correct file from the official **debian.org** website.

Now that you have your installation image, you can execute the installation script (S2.03-lance-installation). If you want more information on how it works, you may take a look at the **appendix at the end of this guide**.

II.2. Installation

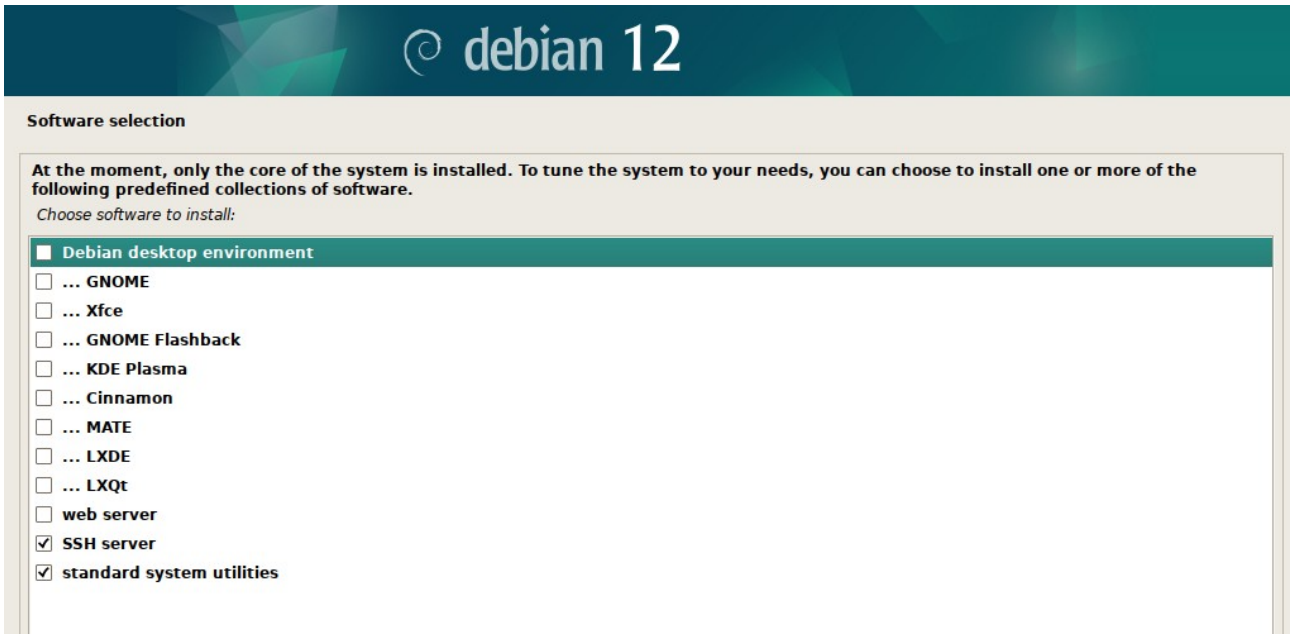
Upon executing the script, the virtual machine will pop up and prompt you to begin the installation. Follow the instructions, and choose the default answers when there isn't an indication.

- Language : English
- Location : other/Europe/France
- Locales : United States, en_US.UTF-8
- Keyboard : French (unless you are more familiar with a different keyboard style)
- **Hostname** : server-YOURLOGIN (for example: server-gentilq)
- **Root Password** : because this virtual machine is used exclusively for educational purposes, you may use something easy to remember, like « root » for example.
- User Account - **Full Name** : your full name, such as « Quentin Gentil »
- **User Name** : your UGA login (following the examples from above, it would be « gentilq »)



- **User Password** : again, use a simple password, like « etu »
- Partition disks : Guided - use entire disk
- Partition disks : All files in one partition

- Partition disks : Yes
- Software Selection : make sure « Debian desktop » is NOT selected and that « ssh server » IS selected



- Install GRUB : Yes
- **Device for boot loader : /dev/sda**

Once the installation is finished, the virtual machine will reboot. Notice that it has no graphical interface. If it does, you should restart the installation and make sure you do not select « Debian desktop ».

➢ # systemctl poweroff

You will use this command (or simply « poweroff ») each time you want to turn off the virtual machine, rather than simply closing the Qemu window.

Finally, on your main machine, use the script « S2.03-déplace-image-disque-sur-erebus4 », that way you can access it easily using the provided script : « S2.03-lance-machine-virtuelle ». Once again, the explainations can be found in the appendix.

To test if you can access the internet, you can try installing a small package, such as « sl » : simply log into the root user with «$ su » then « apt install sl »

To check if the installation went correctly, you can check the /etc/fstab file :

```
gentilq@server-gentilq:~$ 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=aa137c9c-e32a-40c9-8d76-85f3f6e04170 /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=3991c033-4160-49ca-b66f-f5938885e675 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
gentilq@server-gentilq:~$ _
```

III. Installing packages

Let’s install everything needed for the server. To do that, switch to the root user and install the following packages :

III.1. Apache2

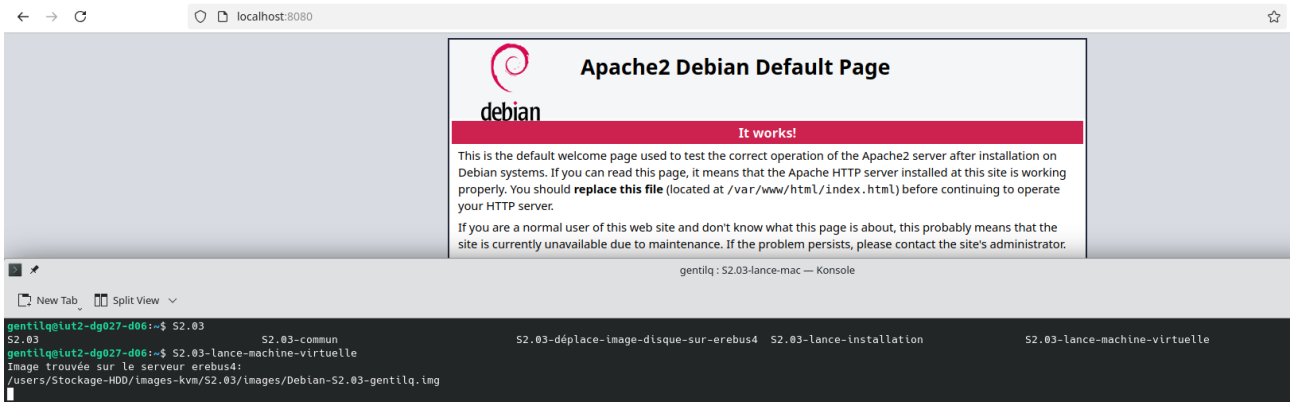
apt install apache2

Before proceeding, you should check if Apache has booted up with the following :

systemctl status apache2

If it is not running, start it : # systemctl start apache2

Then to see if it works properly, try accessing the web page http://localhost:8080 on the **host machine**. It should look something like this :



III.2. PostgreSQL

#apt install postgresql

To check if the installation worked properly, just switch to the « postgres » user (which was created by PostgreSQL) using the su command (# su – postgres), and then taking a look at the default databeses with psql -l (they should be named template0, template1 and postgres).

III.3. PHP and PhpPgAdmin

#apt install php-common libapache2-mod-php php-cli

This installs PHP for Apache2. In order to install PhpPgAdmin, it's a bit more tricky :

- As root, go to the « /etc/apt/sources.list.d/ » folder. If it doesn't exist, create it.
- Use nano and paste the following line :
  deb http://deb.debian.org/debian bookworm-backports main
- Save the file with any name you want, you need to give it the « .list » extension. What this file does is allow apt to install packages from the bookworm-backports repostory.
- Use apt update, apt upgrade and apt clean to finish setting it up.
- And then you can finally install PhpPgAdmin :
  # apt install phppgadmin/bookworm-backports

At the end of this section, type this command. You should have a similar result :

systemctl status apache2 ssh postgresql

```
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-04 08:30:26 CEST; 10min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 477 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 533 (apache2)
      Tasks: 6 (limit: 4642)
     Memory: 23.0M
        CPU: 154ms
     CGroup: /system.slice/apache2.service
             ├─533 /usr/sbin/apache2 -k start
             ├─535 /usr/sbin/apache2 -k start
             ├─536 /usr/sbin/apache2 -k start
             ├─538 /usr/sbin/apache2 -k start
             ├─539 /usr/sbin/apache2 -k start
             └─542 /usr/sbin/apache2 -k start

Apr 04 08:30:25 server-gentilq systemd[1]: Starting apache2.service - The Apache HTTP Server...
Apr 04 08:30:26 server-gentilq apachectl[509]: AH00558: apache2: Could not reliably determine the server's fully qualified doma>
Apr 04 08:30:26 server-gentilq systemd[1]: Started apache2.service - The Apache HTTP Server.

● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-04 08:30:26 CEST; 10min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 489 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 516 (sshd)
      Tasks: 1 (limit: 4642)
     Memory: 6.4M
        CPU: 42ms
     CGroup: /system.slice/ssh.service
             └─516 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Apr 04 08:30:25 server-gentilq systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Apr 04 08:30:26 server-gentilq sshd[516]: Server listening on 0.0.0.0 port 22.
Apr 04 08:30:26 server-gentilq sshd[516]: Server listening on :: port 22.
Apr 04 08:30:26 server-gentilq systemd[1]: Started ssh.service - OpenBSD Secure Shell server.

● postgresql@15-main.service - PostgreSQL Cluster 15-main
     Loaded: loaded (/lib/systemd/system/postgresql@.service; enabled-runtime; preset: enabled)
     Active: active (running) since Fri 2025-04-04 08:30:29 CEST; 10min ago
    Process: 478 ExecStart=/usr/bin/pg_ctlcluster --skip-systemctl-redirect 15-main start (code=exited, status=0/SUCCESS)
   Main PID: 534 (postgres)
      Tasks: 6 (limit: 4642)
     Memory: 39.6M
        CPU: 492ms
lines 1-47/57 79%
```

(Note that Apache, SSH and PostgreSQL are all active and running on the server.)

IV. Setting up and testing

IV.1. PostgreSQL

Let's create a database and configure it so you can access it from the host machine :

First, connect to the postgres user :

```
# su - postgres
```

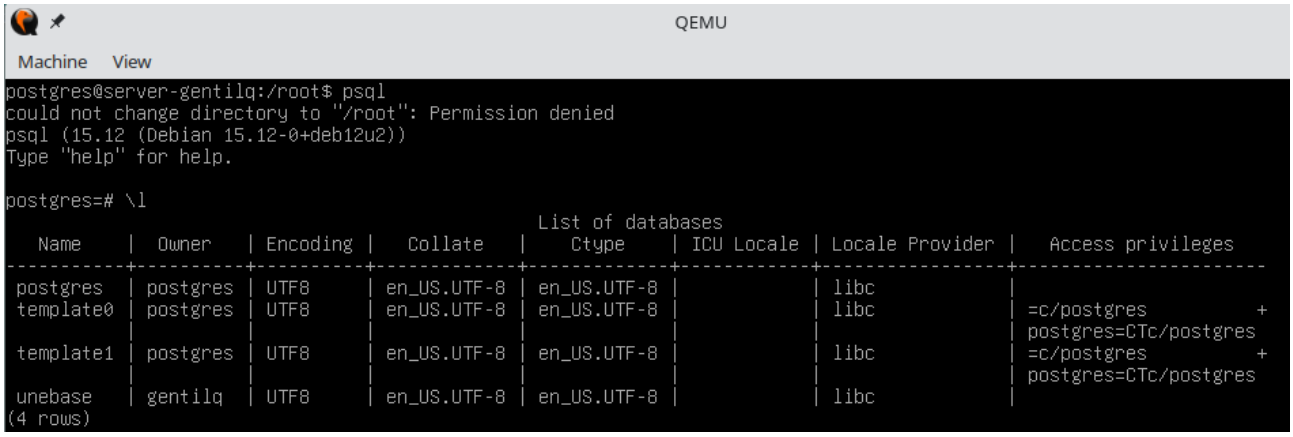
Then connect to PostgreSQL :

```
$ psql
```

Since we want to set it up so it is accessible from the host machine, let's create a user in PostgreSQL with your UGA login, and then a database that this user will own :

```
CREATE USER yourUGAlogin ;
CREATE DATABASE yourDatabaseName WITH owner=yourUGAlogin ;
```

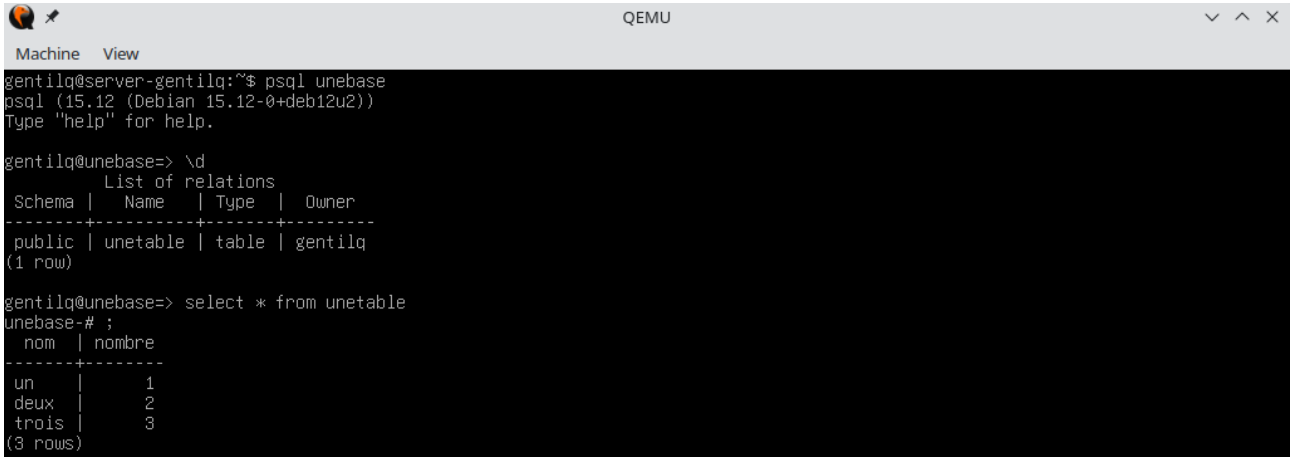
There should now be 4 databases : the three default ones and the one you just made. Notice how the « Owner » column correctly labels the user you just chose :



Right now, you're logged as the psql superadmin user « postgres », which can access this database because it has superadmin privileges. Since you have a user with your UGA login on the virtual machine, you'll also be able to connect from that one. Switch to that user by typing « exit » until you've logged out of both the postgres and root users, then try logging on the database :

```
$ psql yourDatabaseName
```

Because this user is the owner of the database, you can freely create tables, add rows and edit permissions freely :



So now that you have created a database, we can set it up so you can access it from the host machine by tweaking some values in the PostgreSQL configuration files :

Switch to the root user : $ su -

Type the following command : # nano /etc/postgresql/15/main/postgresql.conf

This file contains a lot of general settings for configuring your PSQL server.
Since we only want to access it from the host machine, you just have to change this line :

#listen_addresses = 'localhost'

and change it into this line :

listen_addresses = '*'

(so basically remove the hash symbol and replace « localhost » with « * »)

Then you'll want to edit this other file: # nano /etc/postgresql/15/main/pg_hba.conf

This one is for client authentication : you can configure it to allow specific users on specific ip adresses to access specific databases with certain specifications.

Again, we keep it simple here, and all you have to do is find the following line :

host all all 127.0.0.1/24 scram-sha-256

and change it to :

host all all 0.0.0.0/0 scram-sha-256

And that's it ! As long as the server is running, you can now access the database from the host machine :

$ psql -h localhost yourDatabaseName

Once again, the user you're using is the owner of the database, so you can so anything in there.

```
gentilq@iut2-dg027-d06:~$ psql -h localhost unebase
Password for user gentilq:
psql (15.12 (Debian 15.12-0+deb12u2))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

gentilq@unebase=> \d
          List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+---------
 public | unetable | table | gentilq
(1 row)

gentilq@unebase=> select * from unetable ;
  nom  | nombre
-------+--------
 un    |      1
 deux  |      2
 trois |      3
(3 rows)

gentilq@unebase=>
```

```
gentilq : S2.03-lance-mac — Konsole
New Tab   Split View
gentilq@iut2-dg027-d06:~$ S2.03
S2.03                                   S2.03-commun                    S2.03-déplace-image-disque-sur-erebus4  S2.03-lance-installation        S2.03-lance-machine-virtuelle
gentilq@iut2-dg027-d06:~$ S2.03-lance-machine-virtuelle
Image trouvée sur le serveur erebus4:
/users/Stockage-HDD/images-kvm/S2.03/images/Debian-S2.03-gentilq.img
```

However, that also means that anyone can log into this database by simply using your login. As such, you should add a password with the \password command.

This password is encrypted and stored in the pg_shadow table of psql. You can see its contents by logging in as « postgres » and typing in SELECT * FROM pg_shadow :

```
QEMU
Machine   View
postgres@server-gentilq:/root$ psql
could not change directory to "/root": Permission denied
psql (15.12 (Debian 15.12-0+deb12u2))
Type "help" for help.

postgres=# select * from pg_shadow
postgres-# ;
 usename  | usesysid | usecreatedb | usesuper | userepl | usebypassrls |
        passwd                                                        | valuntil | useconfig
----------+----------+-------------+----------+---------+--------------+-----------------------------------------------------------
-----------------------------------------------------------------------+----------+-----------
 postgres |       10 | t           | t        | t       | t            |
                                                                       |          |
 gentilq  |    16389 | f           | t        | f       | f            | SCRAM-SHA-256$4096:W2sTW/nE//OxUbJLMU9tNg==$YUmhEp5c0kU
s+vbklenatqo9D462youlyh4JdaBneKs=:9efnHja3j1/y724N+TxR//o3dXDz6XUzt7hVShP7FZU= |          |
(2 rows)

(END)
```

As you can see, the password is encrypted with SHA-256, which happens to be a specification that we previously set in the pg_hba.conf file (scram-sha-256)

IV.2 PHP

To test the PHP installation, let's try accessing a .php file from the host machine :

- From the **host machine**, copy the provided .php file to the virtual machine :
  $ scp -P 2222 (filepath)/file.php yourUGAlogin@localhost:/tmp/
- Then from the **virtual machine**, you'll want to put it in the /var/www/html folder :
  $ su -
  # mv /tmp/file.php /var/www/html
- You can now try accessing it from the host machine with any browser with the url « http://localhost:8080/file.php », and it should look like this :

localhost:8080/page_sae_S2.03.php

Bonjour

Je suis www-data

Qui est connecté ?

```
gentilq  tty1         Apr  8 10:26
```

Mes disques sont

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 noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s2:  mtu 1500 qdisc fq_codel 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 86280sec preferred_lft 86280sec
    inet6 fec0::5054:ff:fe12:3456/64 scope site dynamic mngtmpaddr
       valid_lft 86280sec preferred_lft 14280sec
    inet6 fe80::5054:ff:fe12:3456/64 scope link
       valid_lft forever preferred_lft forever
```

My apache install is

```
ii  apache2                      2.4.62-1~deb12u2          amd64        Apache HTTP Server
ii  apache2-bin                  2.4.62-1~deb12u2          amd64        Apache HTTP Server (modules and other binary files)
ii  apache2-data                 2.4.62-1~deb12u2          all          Apache HTTP Server (common files)
ii  apache2-utils                2.4.62-1~deb12u2          amd64        Apache HTTP Server (utility programs for web servers)
ii  libapache2-mod-php           2:8.2+93                  all          server-side, HTML-embedded scripting language (Apache 2 module) (default)
ii  libapache2-mod-php8.2        8.2.28-1~deb12u1          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; preset: enabled)
     Active: active (running) since Tue 2025-04-08 10:25:09 CEST; 2min 0s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 468 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 542 (apache2)
      Tasks: 9 (limit: 4642)
     Memory: 28.2M
        CPU: 177ms
     CGroup: /system.slice/apache2.service
             |-542 /usr/sbin/apache2 -k start
             |-545 /usr/sbin/apache2 -k start
             |-546 /usr/sbin/apache2 -k start
             |-547 /usr/sbin/apache2 -k start
             |-548 /usr/sbin/apache2 -k start
             |-549 /usr/sbin/apache2 -k start
             |-626 /usr/sbin/apache2 -k start
             |-638 sh -c "systemctl status apache2"
             `-639 systemctl status apache2
```

My postgresql install is

```
ii  postgresql                    15+248                          all          object-relational SQL database (supported version)
ii  postgresql-15                 15.12-0+deb12u2                 amd64        The World's Most Advanced Open Source Relational Database
ii  postgresql-client-15          15.12-0+deb12u2                 amd64        front-end programs for PostgreSQL 15
ii  postgresql-client-common      248                             all          manager for multiple PostgreSQL client versions
ii  postgresql-common             248                             all          PostgreSQL database-cluster manager
```

My postgresql status is

```
* postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since Tue 2025-04-08 10:25:12 CEST; 1min 57s ago
    Process: 603 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 603 (code=exited, status=0/SUCCESS)
        CPU: 627us
```

My ssh install is

```
ii  libssh2-1:amd64               1.10.0-3+b1                     amd64        SSH2 client-side library
ii  openssh-client                1:9.2p1-2+deb12u5               amd64        secure shell (SSH) client, for secure access to remote machines
ii  openssh-server                1:9.2p1-2+deb12u5               amd64        secure shell (SSH) server, for secure access from remote machines
ii  openssh-sftp-server           1:9.2p1-2+deb12u5               amd64        secure shell (SSH) sftp server module, for SFTP access from remote machines
ii  task-ssh-server               3.73                            all          SSH server
```

My ssh status is

```
* ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-04-08 10:25:09 CEST; 2min 0s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 475 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 525 (sshd)
      Tasks: 1 (limit: 4642)
     Memory: 6.4M
        CPU: 40ms
     CGroup: /system.slice/ssh.service
             `-525 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

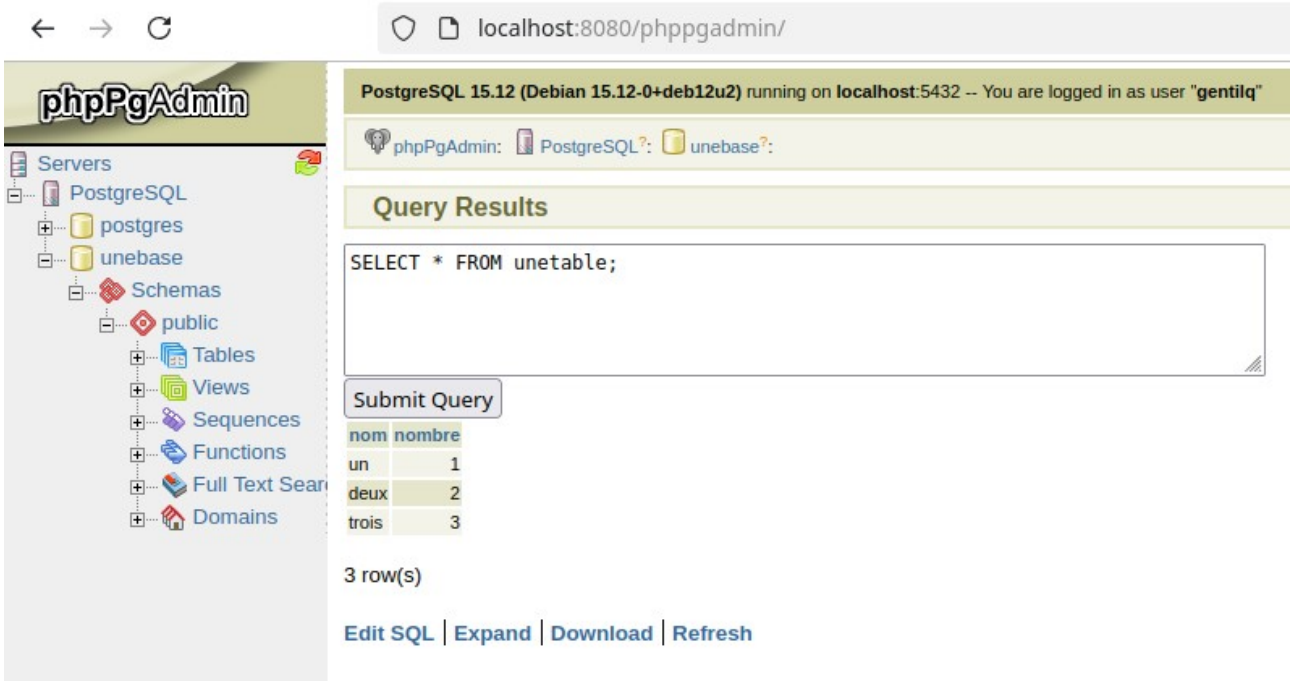
This php file displays information on the server’s Apache, PostgreSQL and SSH installs, as well as the result of the « $ ip addr » command (which lets you see the server’s interfaces).

IV.3. PhpPgAdmin

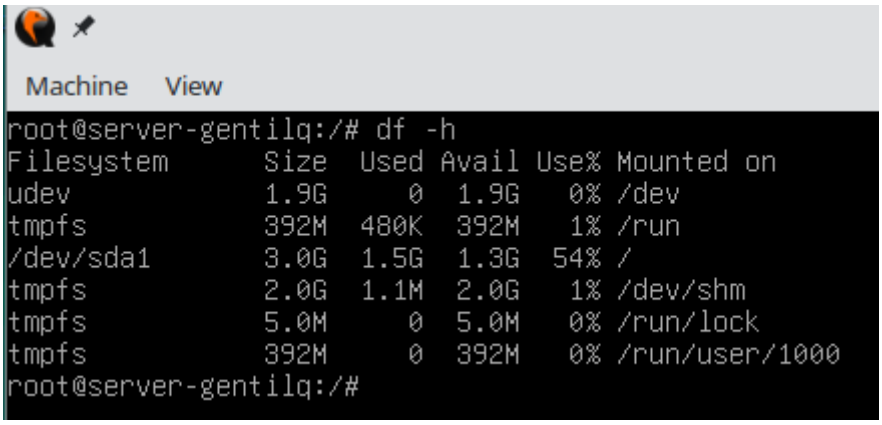
PhpPgAdmin is a web application that allows you to administrate your PostgreSQL database from your browser. It is supposed to be set up correctly when installed, but because we're running Postgres 15, we'll have to change one line in the configuration file :

- # nano /usr/local/www/phpPgAdmin/classes/database/Connection.php
- Find the line that starts with « case '14' : [...] » and replace « 14 » by « 15 ».

You can now access PhpPgAdmin from the **host machine** on a browser with the url « http://localhost:8080/phppgadmin ». Just log into your user and you can now modify your databases from your browser !



And with that, you have finished setting up your server !



V. Appendix

In order to create your virtual machine, along with your installation image, you'll need a disk image. To create it, use this command :

```
$ qemu-img create yourDiskImage.img 4G
```

Now, with both your installation and disk images ready, you can create it :

```
$ qemu-system-x86_64 -machine q35 -cpu host -m 4G -enable-kvm -device VGA,xres=1024,yres=768 -display gtk,zoom-to-fit=off -drive yourDiskImage.img -device e1000,netdev=net0 -netdev user,id=net0,hostfwd=tcp::2222-:22,hostfwd=tcp::4443-:443,hostfwd=tcp::8080-:80,hostfwd=tcp::5432-:5432 -cdrom yourInstallationImage.iso
```

Let's take a quick glance at what this all means :

- -machine q35 is the chipset model ;
- -cpu host means the host machine's cpu will be used to run the vm ;
- -m 4G specifies that 4GB of RAM are allocated to the vm ;
- -enable-kvm enables KVM acceleration ;
- -device VGA,xres=1024,yres=768 adds a video card on VGA with the specified resolution ;
- -display gtk,zoom-to-fit=off is for the way the window will be displayed ;
- -drive yourDiskImage.img is to pick the primary hard disk for the vm ;
- -device adds a network card (here an E1000 connected to net0) ;
- -netdev redirects host ports for the virtual machine:
  - Host port 2222 → VM port 22
  - Host port 4443 → VM port 443
  - Host port 8080 → VM port 80
  - Host port 5432 → VM port 5432
- -cdrom yourInstallationImage.iso makes the vm interpret your .iso file as a CD-ROM, which will allow you to install the OS contained in it (Debian 12).

After installing, if you want to simply boot the virtual machine, use that same command without the final -cdrom parameter.