Debian 12 server with Apache, PostgreSQL and PHP installation guide

Author:
MORAS Evan

Installation guide to get a Debian 12 Server

This manual will teach you how to install and configure a server on a virtual machine, including services such as Apache, PostgreSQL and SSH. There you'll find step-by-step instructions and screenshots for each step, as well as tips for avoiding common mistakes and improving your server security.

Summary

CHAPTER 0

WELCOME !

You have decided to install and configure a server on a virtual machine. You are in the right place! This manual has been specially written to meet your expectations as simply as possible!

You may be wondering if server installation is complex. I understand. Bad configurations happen often. You build your server, follow a few tutorials, and everything can go wrong. A few hours later, you have an unusable server, and it's a disaster.

The best way to learn something is to research it and learn from experienced people. That's why you're here. You want a guide that clearly explains how to install and configure a server efficiently and securely.

This guide will teach you how to install and configure a server, including services like Apache, PostgreSQL, and SSH. From planning to installation and configuration. You'll also find tips for the security of your installation.

CHAPTER 1

INSTALLING DEBIAN 12 SERVER

1) Downloading and Verifying the ISO Image:

To install Debian, we will first need ISO images, or disk images, which will allow you to install all types of things on your PC, like Windows 10, for example, and here, it is Debian.

The ISO image and files to verify its integrity are available here:
https://cdimage.debian.org/cdimage/release/current/amd64/iso-cd/

The file that interests us is the one at the bottom of the page named "SHA512SUMS".

To save time, the IUT has already downloaded these files and can be found in the directory «/usr/local/images-ISO/» .

Next, you have to you have to check the integrity, so just type in a terminal the command "sha512sum" then the path that I gave you just before.

And you just have to check if you have same lines in the document «SHA512SUMS» from the website and with the command you did just before.

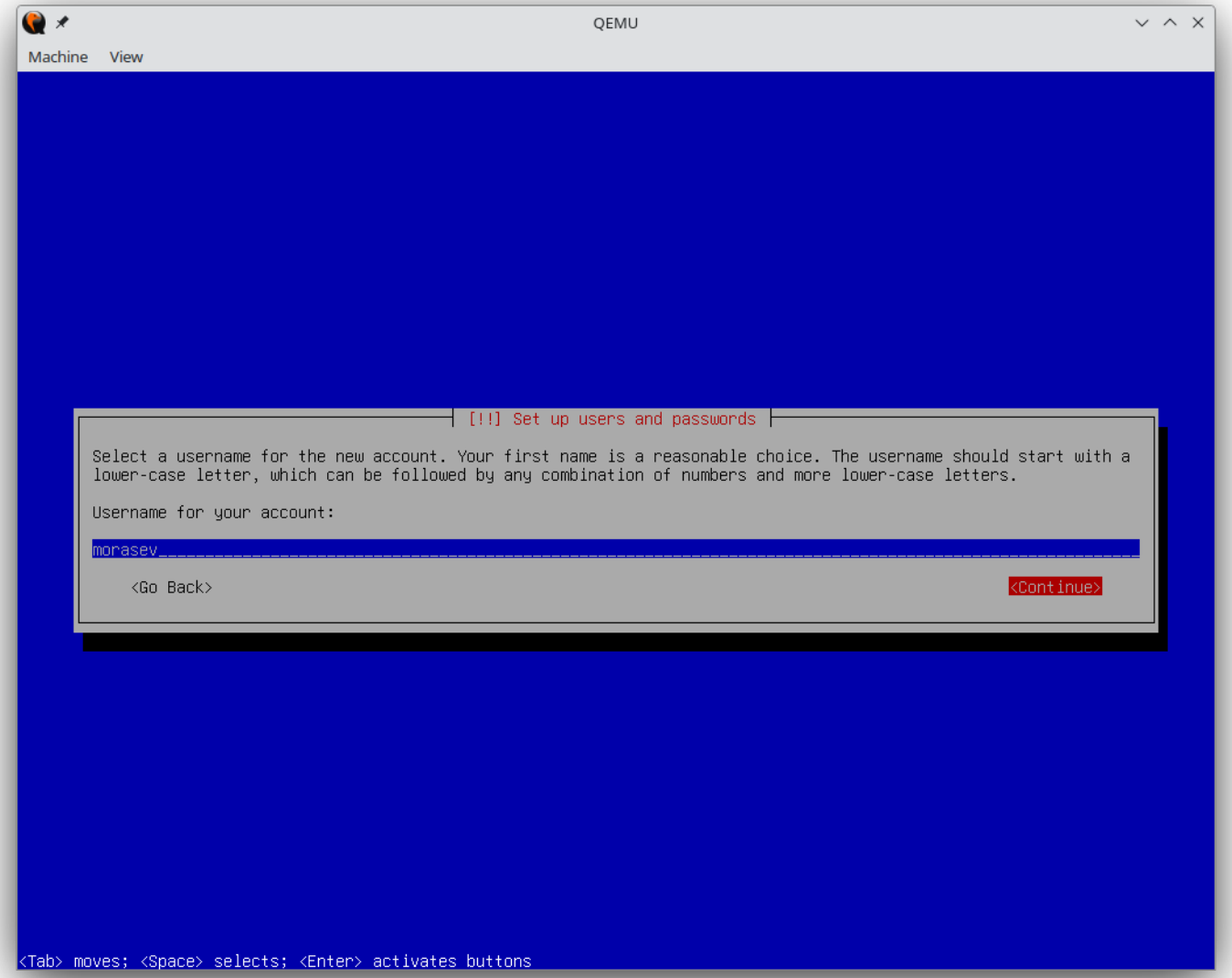
2) Boot the Machine to the ISO Image:

Now that the ISO files are verified, just go into a terminal and type the command "S2.03-launch-installation". This command will just boot the current computer on the installation ISO image and therefore a window will open.

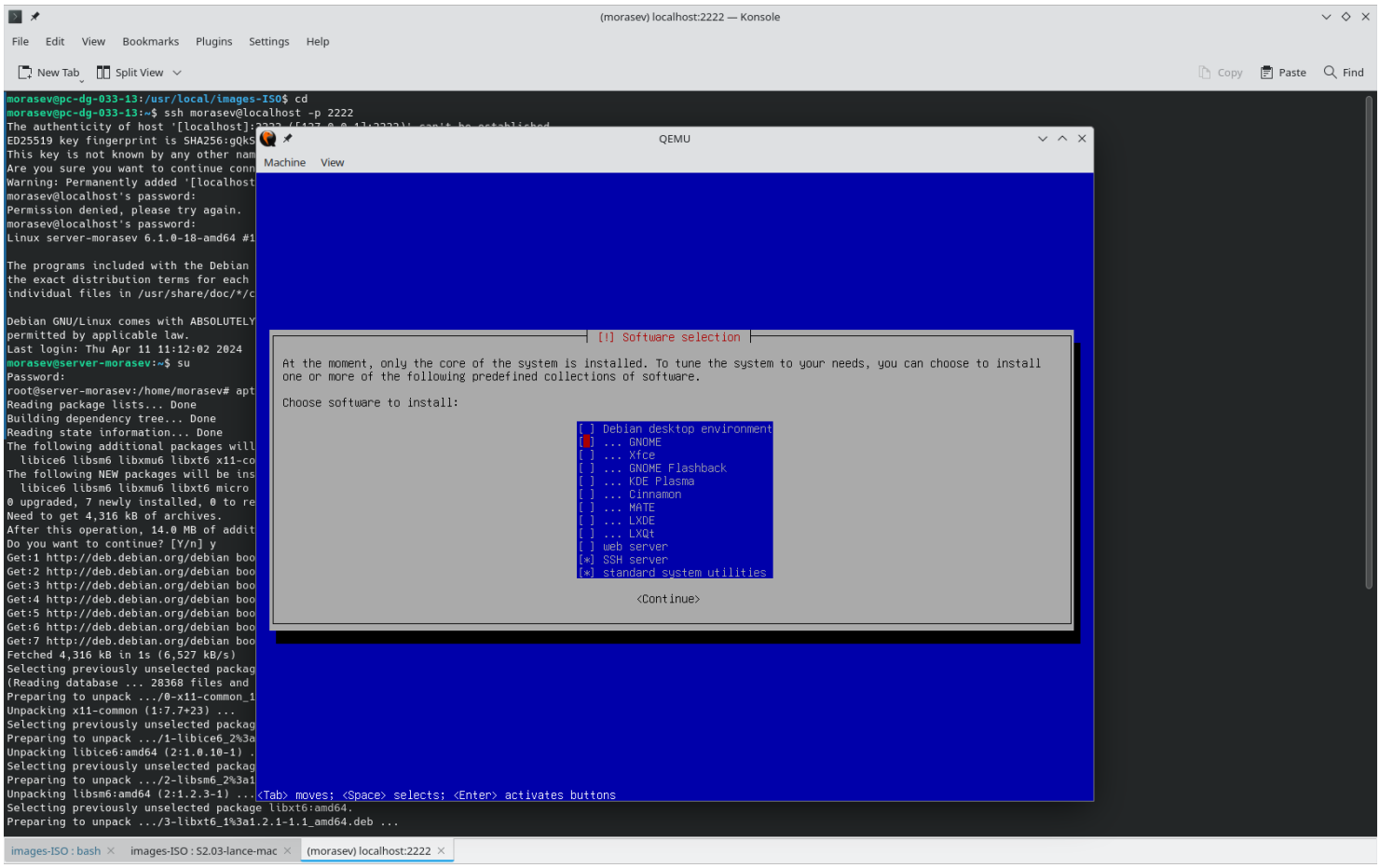
3) Installation Configuration:

Now it's time to start configuring our virtual machine, each written step will be important, so it's very important to follow carefully and do as indicated in this guide. When nothing is specified, make the choice proposed by default. Tip: The most crucial choice is to install Debian without a GUI.

1. **Language:** English
2. **Location :** other / Europe / France
3. **Locales :** United States, en_US.UTF-8
4. **Keyboard :** French
5. **Hostname :** use server-«YOUR_LOGIN_UGA»
6. **Root Password :** use a simple password, like «root» for example. In this context, this does not pose a security problem. Check the «Show Password» box to be sure that the password entered is the one you want.
7. **User Account – Full Name :** Your full name, for example «Jean Toto».
8. **User Name :** Enter your uga login name. Just below a photo showing an example, you should have the same thing :



9. **User Password :** enter a simple password, for example "etu". Check the “Show Password” box to be sure that the password entered is the one you want.
10. **Partition disks :** Guided – use entire disk.
11. **Partition disks :** All files in one partition.
12. **Partition disks :** Yes.
13. **Software Selection :** check that "Debian desktop" is not checked and that "ssh server is checked. Just below a photo showing an example, you should have the same thing :



14. **Install GRUB :** Yes.
15. **Device for boot loader :** /dev/sda

Once the installation is complete, the virtual machine restarts. Once on it, you must turn it off, either by typing the "poweroff" command, or by clicking at the top left on "Machine" then "power Down".

4) Moving the disk image:

The file was created on the local disk of the Linux station. You must therefore move this image to the erebus4 server to be able to use your virtual machine more easily later. (You can also copy this image to a USB key)
To do that, just type this command on your Linux computer, not your virtual machine: «S2.03-déplace-image-disque-sur-erebus4».

CHAPTER 2

CHECKING DEBIAN SERVER

So that's it, your virtual machine is configured and installed, ready to launch!
To verify that everything is fine on your new VM (Virtual Machine), we will perform some checks.

In first, to launch your VM, just just type the command "S2.03-lance-machine-virtuelle

" in a terminal.

1) Check contact with the outside:

To check the contact of your VM with the outside world, firts, you will type the "ip addr" command and retrieve the Ethernet and IP characteristics of your machine.

Secondly, you will use the command "ping" then the name of a website that you know, for example "ping google.com". It is thanks to this command that you will see if your VM can access the outside world, it has access to the Internet.

2) Checking for Absence of Xorg Server:

Checks for the absence of the Xorg server on your machine (using «dpkg -l | grep xorg»). The machine machine is not equipped with an X server for reasons of performance, security, simplicity and suitability for administrative tasks which are mainly carried out via the command line.

3) Port forwarding and SSH access :

To allow you to access the servers running in your virtual machine from clients running on your Linux station, port redirections have been set up (accessible via the S2.03-common file):

Service réseau	Port de la VM	Port sur la station Linux	Exemple d'utilisation depuis la station Linux
SSH	22	2222	$ ssh toto@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

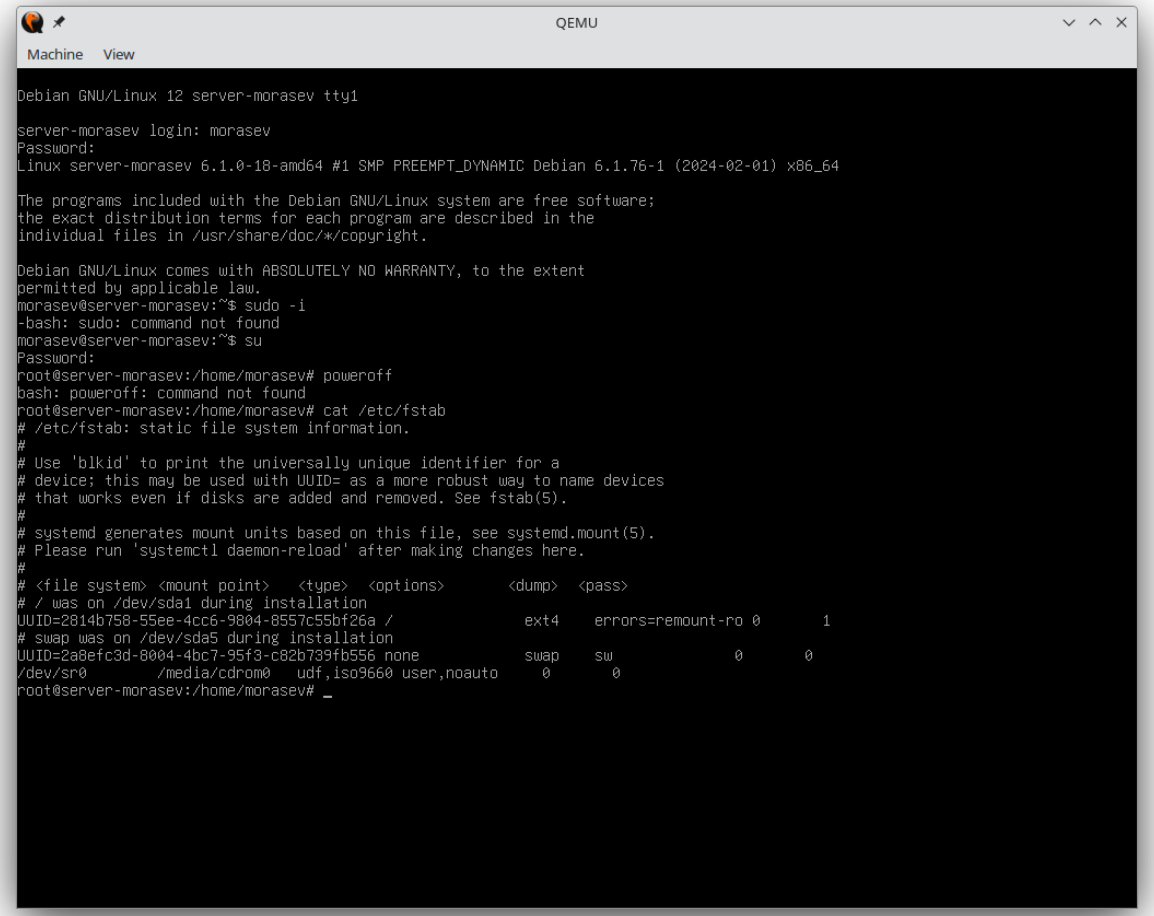
We are going to do a little test, you are going to connect via SSH to your virtual machine, so type the command "ssh login@localhost-p 2222" on your Linux machine.

Then, switch to root account, enter the "su" command and then enter the password you provided during configuration.

And to check that everything works, try installing a Debian package, for example the "micro" text editor. To do this, you just type the command "apt install micro" and it will install.

4) result of the command «cat /etc/fstab»:

For this last verification step, you will just type the command «cat /etc/fstab» in your virtual machine, while being root. Just below a photo showing an example, you should have the same thing :



CHAPTER 3

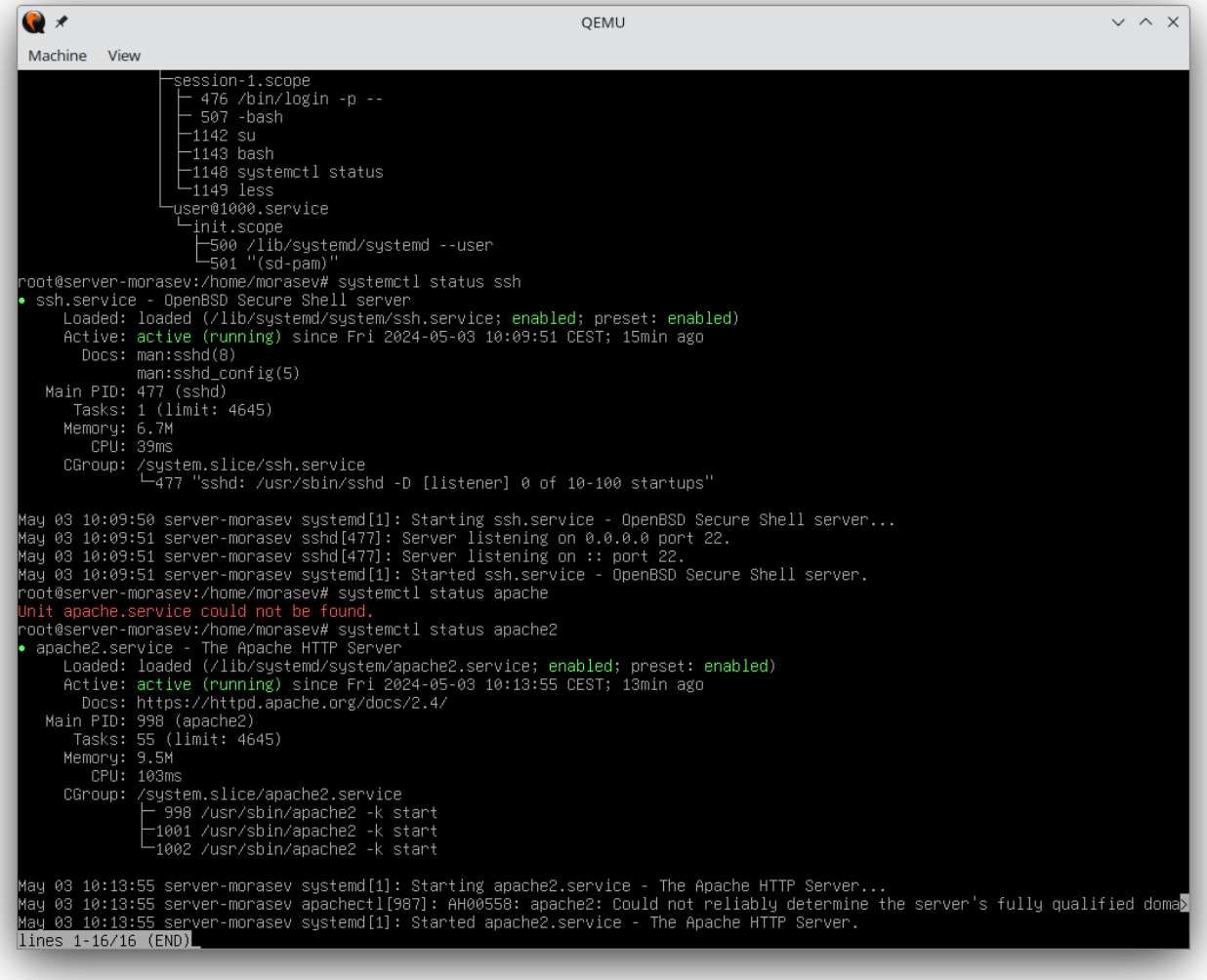
APACHE INSTALLATION

1) Apache Installation:

For this first installation, we will take care of Apache, you will see, it will be simple and quick. Become root on your virtual machine and type the command «apt install apache2».

(If you want to have more information on Apache, you can go to their site and also explain the installation of Apache): https://httpd.apache.org/docs/2.4/en/install.html

Once the installation is complete, to check that Apache is started on your machine, type the command «systemctl status apache2». Just below a photo showing an example, you should have the same thing :



And if Apache is not started, just type this command: «systemctl start apache2»

2) Apache checks:

Since your machine is a server without a graphical interface, it is not possible to display an HTML page. You can connect to the Apache server using telnet and entering the character string «HEAD / HTTP/1.0» followed by two newlines. The server should respond «HTTP/1.1 200 OK» like this:

```
$ telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
[...]
```

Although it is not possible to display a web page on the virtual machine, it is possible to do so from the host machine. To do this, you must redirect a port on the host machine (for example 8080) to port 80 (default port for web servers) on the virtual machine.

So open a web browser on the host Linux machine and navigate to the URL http://localhost:8080/. Check that you are on the default page of the virtual machine's Apache server. Just below a photo showing an example, you should have the same thing:

morasev : S2.03-lar

File Edit View Bookmarks Plugins Settings Help

New Tab Split View

```
morasev@pc-dg-033-13:~$ S2.03-lance-machine-virtuelle
Image trouvée sur le serveur erebus4:
/users/Stockage-HDD/images-kvm/S2.03/images/Debian-S2.03-morasev.img
```

QEMU

Machine View

```
May 03 10:09:50 server-morasev systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
May 03 10:09:51 server-morasev sshd[477]: Server listening on 0.0.0.0 port 22.
May 03 10:09:51 server-morasev sshd[477]: Server listening on :: port 22.
May 03 10:09:51 server-morasev systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
root@server-morasev:/home/morasev# systemctl status apache
Unit apache.service could not be found.
root@server-morasev:/home/morasev# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Fri 2024-05-03 10:13:55 CEST; 13min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 998 (apache2)
      Tasks: 55 (limit: 4645)
     Memory: 9.5M
        CPU: 103ms
     CGroup: /system.slice/apache2.service
             ├─998 /usr/sbin/apache2 -k start
             ├─1001 /usr/sbin/apache2 -k start
             └─1002 /usr/sbin/apache2 -k start

May 03 10:13:55 server-morasev systemd[1]: Starting apache2.service - The Apache HTTP Server...
May 03 10:13:55 server-morasev apachectl[987]: AH00558: apache2: Could not reliably determine the server's fully qualified doma
May 03 10:13:55 server-morasev systemd[1]: Started apache2.service - The Apache HTTP Server.
root@server-morasev:/home/morasev# telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 03 May 2024 08:28:22 GMT
Server: Apache/2.4.57 (Debian)
Last-Modified: Fri, 03 May 2024 08:13:53 GMT
ETag: "29cd-617884ced9e4c"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

Connection closed by foreign host.
```

IUT2 Grenoble | S2.03 - CDC et | S2.03 - Séance | Apache2 Debia | New Tab

localhost:8080

debian

Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

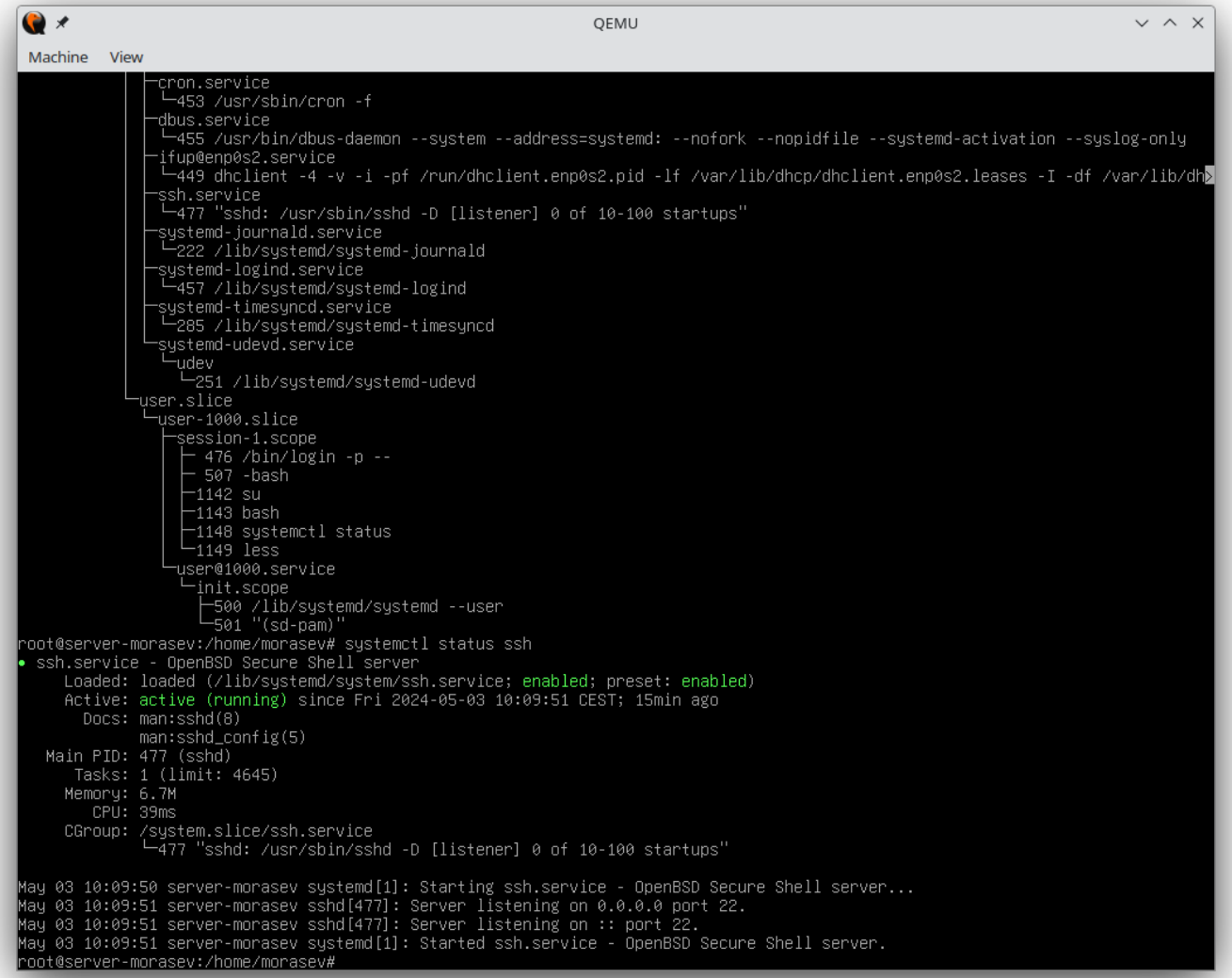
```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration,

Apache2 Debian Default Page: It works — Mozilla Firefox | morasev : S2.03-lance-mac — Konsole | QEMU | 24,7% | 10:32 03/05/2024

For the rest, we will need SSH on this virtual machine, so we have to check his status.

Just type «systemctl status ssh», like this:



CHAPTER 4

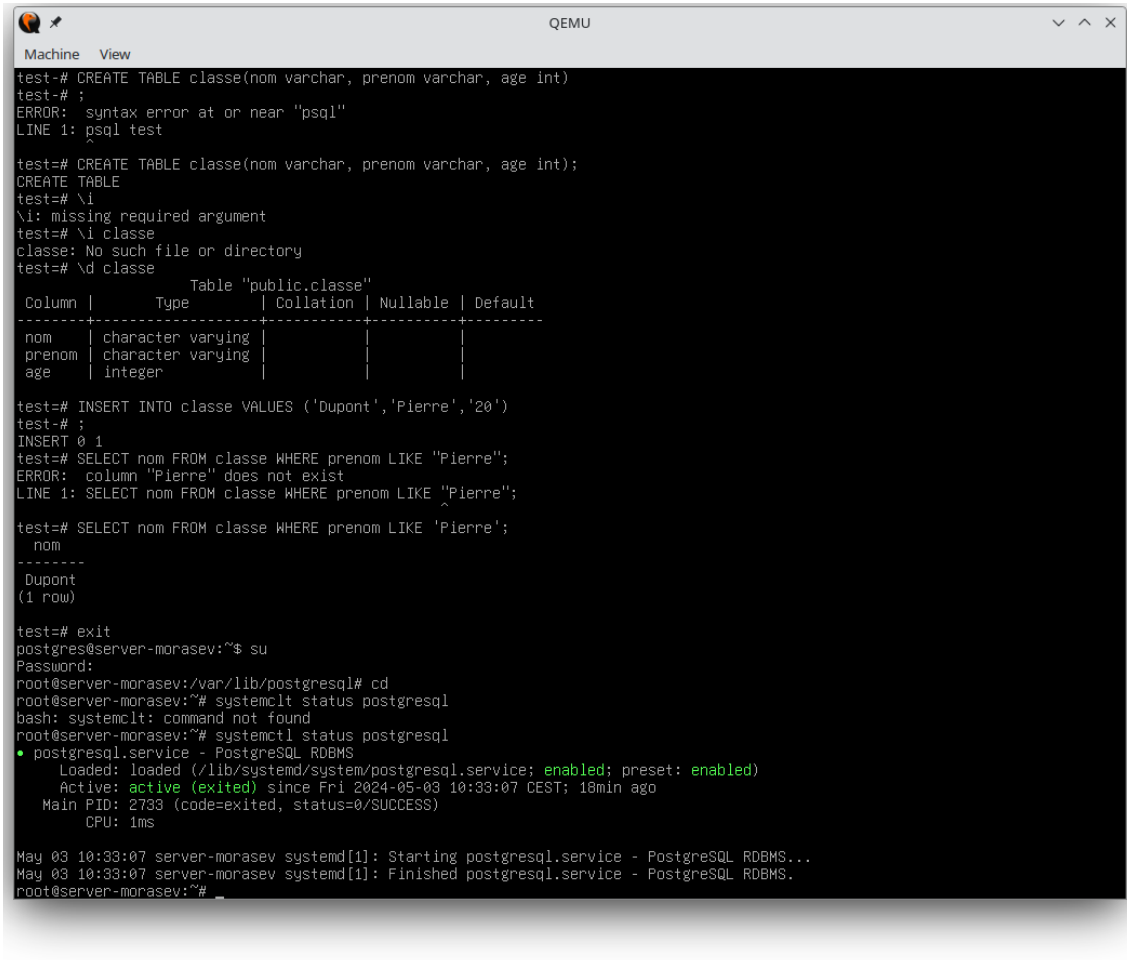
POSTGRESQL INSTALLATION

1) PostgreSQL INSTALLATION:

For the second installation, we will install PostgreSQL. To do this, as with Apache, you must be in root mode. Then you will just type the command «apt install postgresql».

(If you want to have more information on PostgreSQL, you can go to their site and also explain the installation of PostgreSQL): https://www.postgresql.org/

Once the installation is complete, to check that PostgreSQL is started on your machine, type the command «systemctl status postgresql». Just below a photo showing an example, you should have the same thing:



Now, we are going to check the installation, connect under the postgres login using, from the root account, this command : «su – postgres»

And use the command : «psql -l» which will bring up the list of default databases.

2) Using postgresql:

We will now answer questions regarding all the interactions to be done with PostgreSQL.

a) Connect to PostgreSQL from this same shell running on the virtual machine:

pass to postgres user: « su – postgres» and launch the PostgreSQL console: «psql».

b) In PostgreSQL, create a user with your UGA login name:

«CREATE ROLE your_login WITH LOGIN;»

c) Create a database whose owner must be your user:

«CREATE DATABASE database_name WITH OWNER=your_login;»

d) Create a password to access to your database:

Just before, connect with your database with the command «\c database_name»

«\password your_login» and choose a password.

e) Create a simple table:

Then, an example of a simple table : «CREATE TABLE classe(nom varchar, prenom varchar, age int);»

You can visualize your table with «\d classe».

f) Insert a few rows of data into the table:

«INSERT INTO classe VALUES (‘Dupont’,’Pierre’,’20’);»

g) Configure PostgreSQL so that it is accessible from your Linux station. To do this, you will have to modify 2 configuration files, then restart the PostgreSQL server.

The two useful files will be the files «postgresql.conf» and «pg_hba.conf».

1) Find it:

To find the exact path of the two files, we will use «find». First, type the command

«find / -name postgresql.conf».

Then, type the command «find / -name pg_hba.conf».

2) Modify it:

To read and modify these two files, we will use "nano". For the first file, type the command

«nano path_you_found/postgresql.conf»

When your are in, find the «listen_addresses» line and edit it, It should be like this in the end:

listen_addresses = '*'

Saves and closes the file. It's Ctrl+O to save and Ctrl+X to exit.

For the second, type the same command:

«nano path_you_found/pg_hba.conf»

When you are in, find the section «IPv4 local connections :» and your row has to be like:

«host all all 0.0.0.0/0 md5»

Saves and closes the file. In Nano, it's Ctrl+O to save and Ctrl+X to exit.

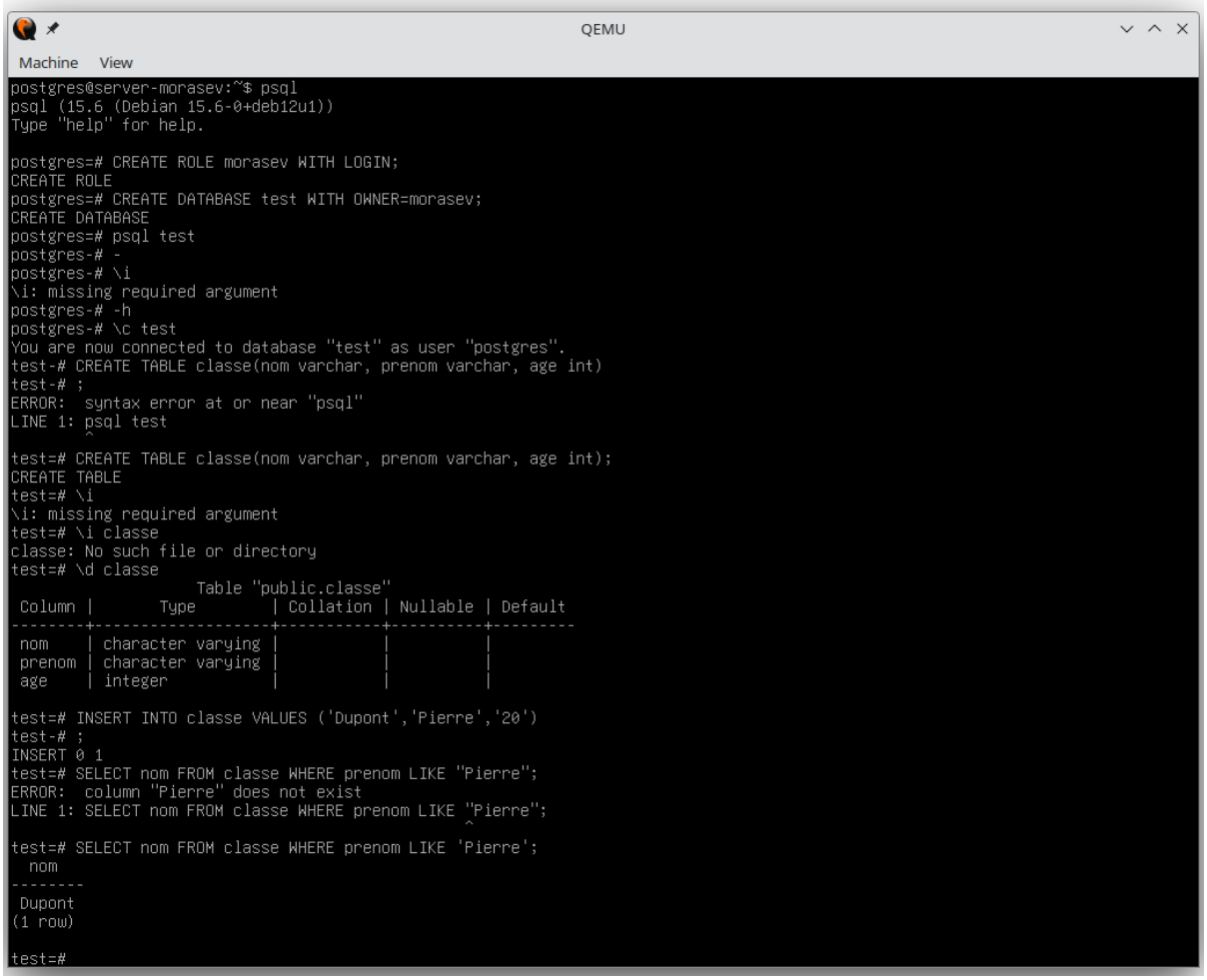
Restart PostgreSQL to apply the changes:

«systemctl restart postgresql»

3) Testing it:

Now that our Linux machine has access to this PostgreSQL, we will test this with a query. We'll start with a simple query on our VM:

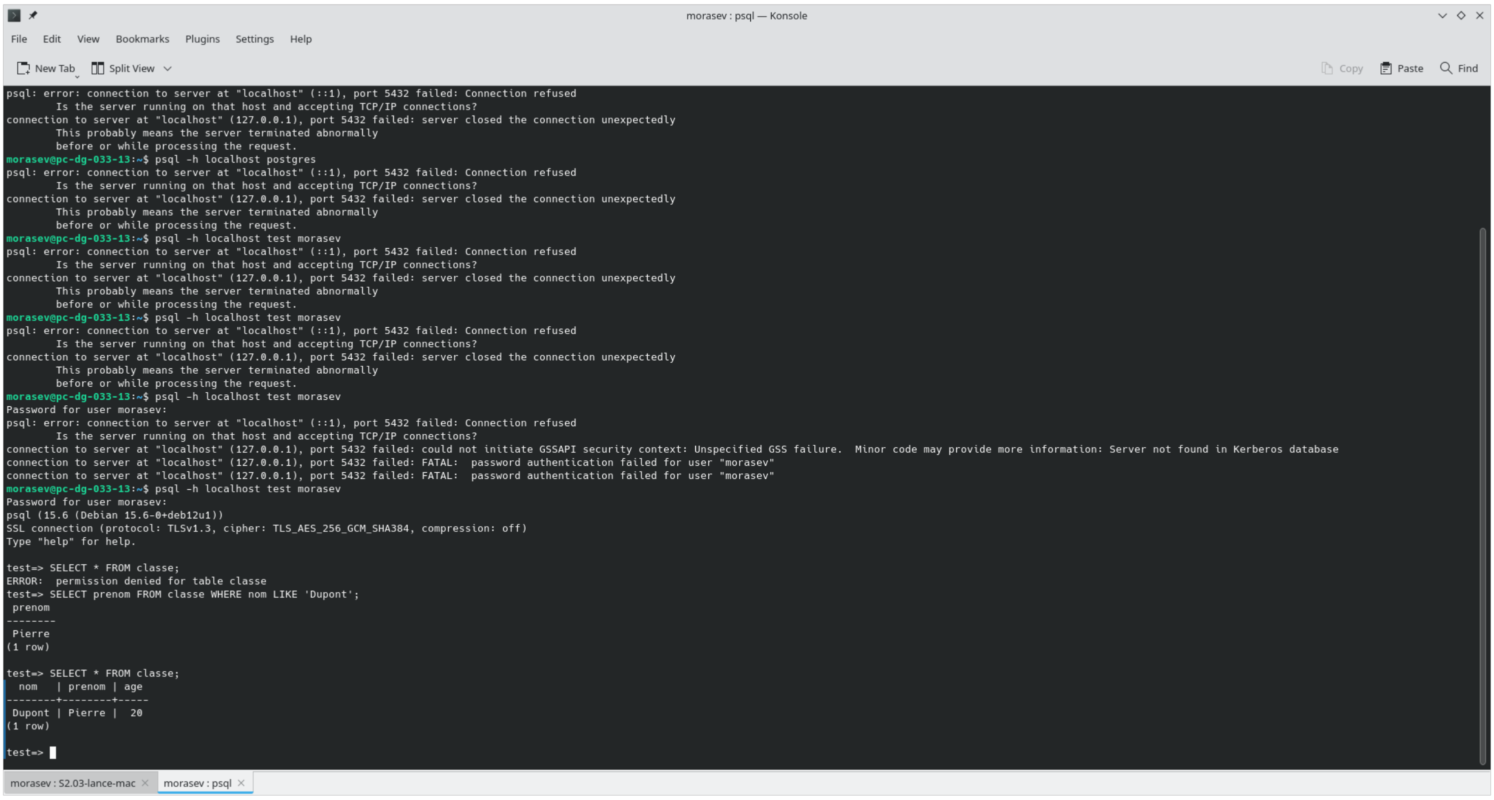
Here, it is, for example, «SELECT nom FROM classe WHERE prenom LIKE ‘Pierre’;»



Then, to write a query from our Linux machine, we must connect to this PostgreSQL, to do this, use this command:

«psql -h localhost -U database_name your_login» and enter the password that you choosed before.

Now, we will write a simple query.
Here, it is, for example, «SELECT * FROM classe;



```
psql: error: connection to server at "localhost" (::1), port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?
connection to server at "localhost" (127.0.0.1), port 5432 failed: server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
morasev@pc-dg-033-13:~$ psql -h localhost postgres
psql: error: connection to server at "localhost" (::1), port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?
connection to server at "localhost" (127.0.0.1), port 5432 failed: server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
morasev@pc-dg-033-13:~$ psql -h localhost test morasev
psql: error: connection to server at "localhost" (::1), port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?
connection to server at "localhost" (127.0.0.1), port 5432 failed: server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
morasev@pc-dg-033-13:~$ psql -h localhost test morasev
psql: error: connection to server at "localhost" (::1), port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?
connection to server at "localhost" (127.0.0.1), port 5432 failed: server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
morasev@pc-dg-033-13:~$ psql -h localhost test morasev
Password for user morasev:
psql: error: connection to server at "localhost" (::1), port 5432 failed: Connection refused
        Is the server running on that host and accepting TCP/IP connections?
connection to server at "localhost" (127.0.0.1), port 5432 failed: could not initiate GSSAPI security context: Unspecified GSS failure.  Minor code may provide more information: Server not found in Kerberos database
connection to server at "localhost" (127.0.0.1), port 5432 failed: FATAL:  password authentication failed for user "morasev"
connection to server at "localhost" (127.0.0.1), port 5432 failed: FATAL:  password authentication failed for user "morasev"
morasev@pc-dg-033-13:~$ psql -h localhost test morasev
Password for user morasev:
psql (15.6 (Debian 15.6-0+deb12u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

test=> SELECT * FROM classe;
ERROR:  permission denied for table classe
test=> SELECT prenom FROM classe WHERE nom LIKE 'Dupont';
 prenom
--------
 Pierre
(1 row)

test=> SELECT * FROM classe;
  nom   | prenom | age
--------+--------+-----
 Dupont | Pierre |  20
(1 row)

test=> 
```

Now, to display the list of PostgreSQL databases with their respective owners. Just type:

«\list». You would have the same thing:

```
QEMU
Machine  View
                                        # (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)


root@server-morasev:~# psql
psql: error: connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: FATAL:  role "root" does not exist
root@server-morasev:~# su -postgres
su: invalid option -- 'o'
Try 'su --help' for more information.
root@server-morasev:~# su - postgres
postgres@server-morasev:~$ psql
psql (15.6 (Debian 15.6-0+deb12u1))
Type "help" for help.

postgres=# \c test
You are now connected to database "test" as user "postgres".
test=# exit
postgres@server-morasev:~$ exit
logout
root@server-morasev:~# \list
bash: list: command not found
root@server-morasev:~# \d
bash: d: command not found
root@server-morasev:~# su - postgres
postgres@server-morasev:~$ psql
psql (15.6 (Debian 15.6-0+deb12u1))
Type "help" for help.

postgres=# \list
                                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    | ICU Locale | Locale Provider |   Access privileges
-----------+----------+----------+-------------+-------------+------------+-----------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +
           |          |          |             |             |            |                 | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +
           |          |          |             |             |            |                 | postgres=CTc/postgres
 test      | morasev  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
(4 rows)

postgres=#
```

To list the contents of the pg_shadow system table and verify that the passwords are hashed with the SHA-256 hash function, you just have to do like that:

with the command «SELECT * FROM pg_shadow;»

QEMU

Machine View

```
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            scram-sha-256
host    replication     all             ::1/128                 scram-sha-256



root@server-morasev:~# service postgresql restart
root@server-morasev:~# su - postgres
postgres@server-morasev:~$ psql
psql (15.6 (Debian 15.6-0+deb12u1))
Type "help" for help.

postgres=# \c test
You are now connected to database "test" as user "postgres".
test=# password morasev
test-# ;
ERROR:  syntax error at or near "password"
LINE 1: password morasev
        ^
test=# \password morasev
Enter new password for user "morasev":
Enter it again:
test=# \d
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | classe | table | postgres
(1 row)

test=# GRANT CONNECT ON classe TO morasev;
ERROR:  invalid privilege type CONNECT for relation
test=# GRANT CONNECT ON classe TO morasev
test-# ;
ERROR:  invalid privilege type CONNECT for relation
test=# GRANT SELECT ON classe TO morasev;
GRANT
test=# SELECT * FROM pg_shadow;
 usename  | usesysid | usecreatedb | usesuper | userepl | usebypassrls |
        passwd                                                                | valuntil | useconfig
----------+----------+-------------+----------+---------+--------------+-------------------------------------------------------
-----------------------------------------------------------------------------+----------+-----------
 postgres |       10 | t           | t        | t       | t            |
                                                                              |          |
 morasev  |    16388 | f           | f        | f       | f            | SCRAM-SHA-256$4096:ooxnQcDQxoEWemqAQxS2jw==$dMsxb7gVKOe
DlrQ2uwNotBoq0W+zNUmoJQM5n+t1rUQ=:hIA55xh/vo1PNnESshfbMiMkJbREFTCGBLnGmCUdlXQ= |          |
(2 rows)

test=#
```

CHAPTER 5

PHP AND PHPPGADMIN INSTALLATION

1) PHP and PhpPgAdmin Installation:

First of all, we will install Php. To do this, switch to root mode on your VM and type the command «apt install php-common libapache2-mod-php php-cli».

(Command found on the php site, here is the link if you want):
https://www.php.net/manual/en/install.unix.php

Now, PhP is installed on your VM !

To check if everything is working good, we will create a php file «info.php» in the repertory «/var/www/html/».

To do this, write «nano /var/www/html/info.php» in the good repertory.

Then, complete it with:

```
<?php
phpinfo();
phpinfo(INFO_MODULES);
?>
```

And, on the host machine, go on http://localhost:8080/info.php on internet to check that a page containing the main characteristics of your PHP installation appears.

Secondly, we will install PhpPgAdmin.

Just type the command «apt install phppgadmin»

(If you want to get more informations, just go on the PhpPgAdmin Wikipedia page):
https://fr.wikipedia.org/wiki/PhpPgAdmin

Then, type «nano /etc/apache2/conf-enabled/phppgadmin.conf»

And you just have to comment the line «#Require local».

Then, type «systemctl reload apache2.service» to reload and save the changes.

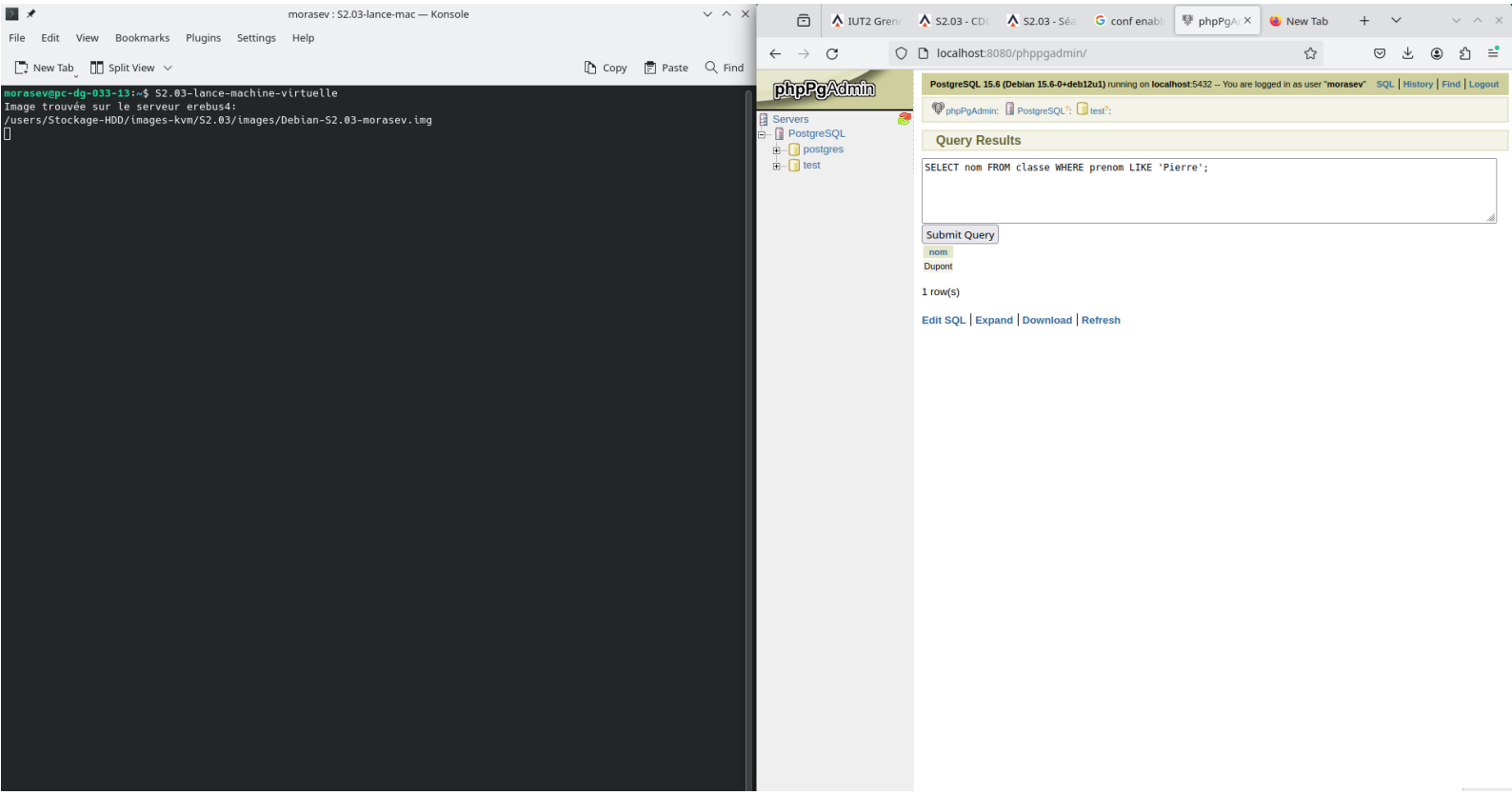
You have now to modify an other file, just do :

«nano /usr/share/phppgadmin/classes/database/Connection.php»

Replace the row «case '14': return 'Postgres';break;» in «case '15': return 'Postgres';break;»

2) Interaction with the database and PhpPgAdmin:

Go on «localhost:8080/phppgadmin/» on the host machine. You will be able to interact with the database just created before. For example:



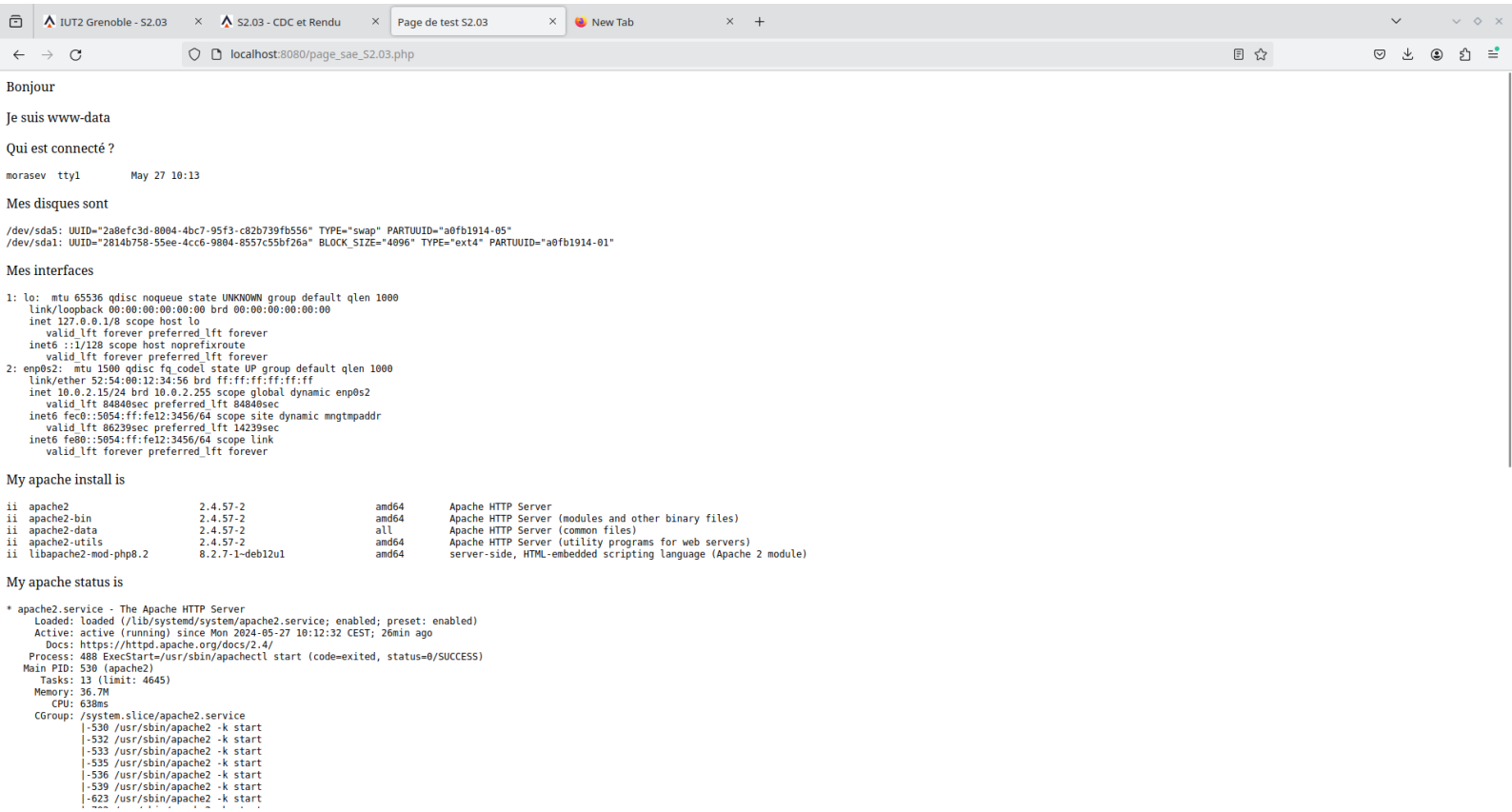
3) Install a php file to the Apache server:

On yout VM, type the command «/sbin/blkid»

Go on the host machine and type this:

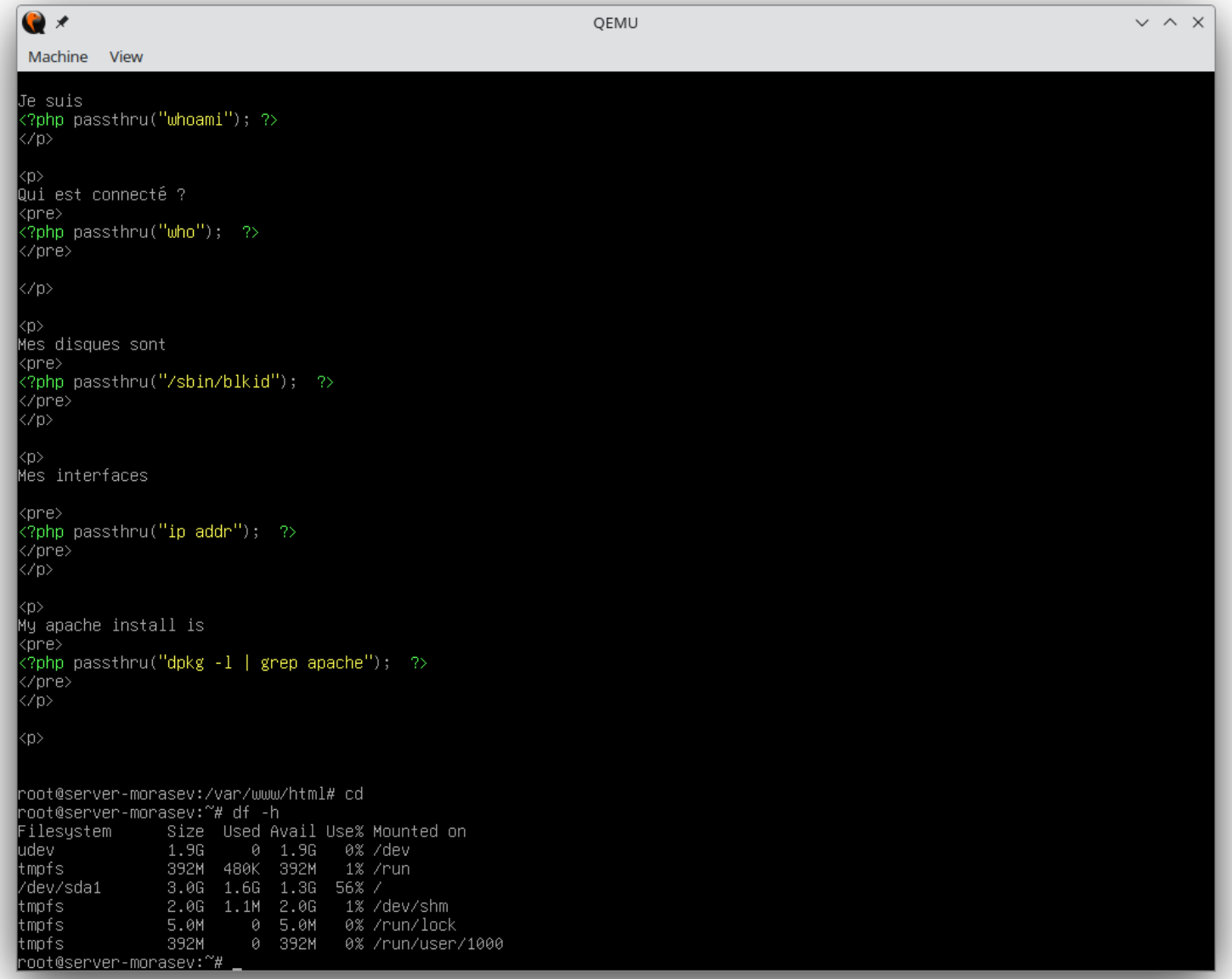
«scp your_login@name_host_machine:/users/info/www/intranet/enseignements/S2.03/page_sae_S2.03.php /var/www/html/» This command will send a php file from the host machine to your VM.

Now, you can go to «localhost:8080/page_sae_S2.03.php» and see the php file. Like this:



Just to finish, Let's see the storage space used and the one remaining in your virtual machine.

With the command «df -h». Like this:



CHAPTER 6

SECURITY ANALYSIS OF YOUR INSTALLATION

The security of your server is essential to protect your data and guarantee the proper functioning of the services. That's why I'm going to give you some advice on how to best secure your new server.

(Everything below has to be done in root mode)

1. Regular updates:

Use «apt update» and «apt upgrade»

You can use too «apt list –upgradable» to see packages that can be updated.

For Apache, make regular updates with «apt upgrade apache2»

For PHP, make regular updates with «apt upgrade php»

For PostgreSQL, make regular updates with «apt upgrade postgreSQL»

2. Configure a firewall:

If you want, you can install a firewall. For example, we will use «ufw» to manage firewall rules. Here's how to allow only the necessary services (SSH, HTTP, HTTPS):

«apt install ufw

ufw allow ssh

ufw allow http

ufw allow https

ufw enable

CHAPTER 7

WEBOGRAPHY

- Debian distribution
- Debian Installation Manuals
- Easy Debian
- Apache Installation Manual
- PostgreSQL
- PHP Installation Manual
- PhpPgAdmin Wikipedia page

So we are at the end of this installation guide! I hope it was clear and helped you a lot! Thanks for reading it!