

Debian 12 Server Installation Guide

Galat Nathanaël

Table of contents

1. Debian 12 installation.....	3
1.1. Preparation for installation.....	3
1.2. Installation.....	3
1.3. Moving the disk image.....	5
1.4. Debian Server Check.....	6
2. Apache and PostgreSQL Installation.....	7
2.1. Apache Installation.....	7
2.2. PostgreSQL Installation.....	8
2.3. Using PostgreSQL on the virtual machine.....	9
2.4. Using PostgreSQL on Linux Machine.....	10
2.5. Database verification.....	11
3. PHP and PhpPgAdmin installation.....	12
3.1. PHP Installation.....	12
3.2. PhpPgAdmin Installation.....	13
4. Security.....	14
5. Command Summary.....	15
6. Installing a virtual machine on a USB key (annex).....	16

1. Debian 12 installation

1.1. Preparation for installation

Please note: Commands with a # in front of them mean they must be run as the root user.

To begin, you need to download the Debian 12 ISO file. However, to save time and disk space, the ISO file has already been downloaded and made available.

To verify its integrity, go to this web page:

<https://cdimage.debian.org/cdimage/release/current/amd64/iso-cd/SHA512SUMS>

Then run the following command:

```
« sha512sum debian-12.10.0-amd64-netinst.iso »
```

Once the ISO has been verified (by comparing the two file codes: debian-12.10.0-amd64-netinst.iso), you can proceed with the installation.

1.2. Installation

Now that you've verified the ISO, you can launch the Debian 12 installation using the "S2.03-lance-installation" command.

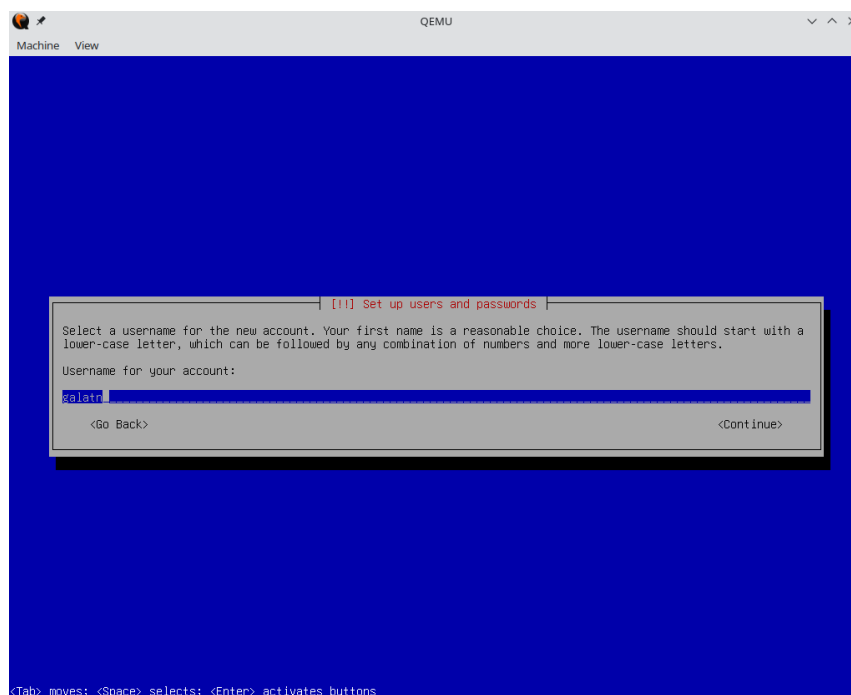
In the "S2.03-commun" script, there are various parameters (options) for installing Debian 12. I'll briefly explain them here:

- ➔ **qemu-system-x86_64** : Launches QEMU for x86_64 (64-bit) architecture.
- ➔ **-machine q35** : Uses the Q35 chipset.
- ➔ **-cpu host** : Uses the same CPU model as the host.
- ➔ **-m 4G** : Allocates 4 GB of RAM to the virtual machine.
- ➔ **-enable-kvm** : Enables hardware acceleration with KVM.
- ➔ **-device VGA,xres=1024,yres=768** : Defines a VGA graphics card with a resolution of 1024x768.
- ➔ **-display gtk,zoom-to-fit=off** : Display with GTK of the VM without automatic zoom.
- ➔ **-drive \$drive** : Specifies a disk image file to use.
- ➔ **-device e1000,netdev=net0** : Adds a virtual Intel e1000 network adapter attached to the net0 interface.

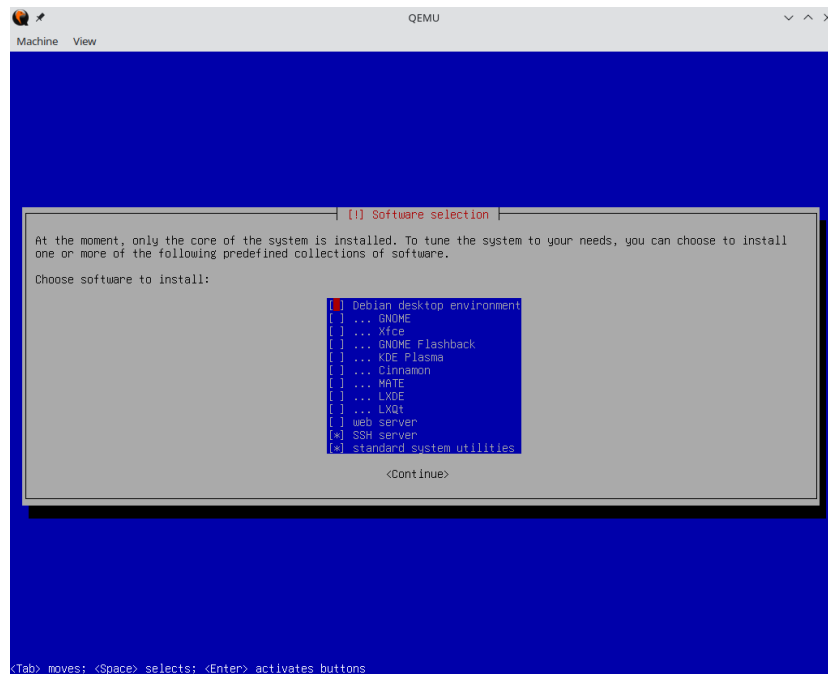
- ➔ **-netdev user,id=net0** : Enables user mode for the network.
- ➔ **hostfwd=tcp::2222-:22** : Redirects port 2222 of the host to the SSH port (22) of the VM.
- ➔ **hostfwd=tcp::4443-:443** : Redirects port 4443 of the host to the HTTPS port (443) of the VM.
- ➔ **hostfwd=tcp::8080-:80** : Redirects port 8080 of the host to the HTTP port (80) of the VM.
- ➔ **hostfwd=tcp::5432-:5432** : Redirects port 5432 of the host to the PostgreSQL port (5432) of the VM.

Now you can proceed with the installation. Caution! Some parts of the installation may need to be modified:

- **Location** : other/Europe/France
- **Keyboard** : French
- **Hostname** : Use server-"VOTRE_LOGIN_UGA"
- **Root Password** : A simple password is recommended, for example "root".
- **User Account - Full Name** : your full name
- **User Name** : enter your UGA login name (as in the image below)



- **User Password** : enter a simple password, for example "etu".
- **Partition disks** : Yes
- **Software Selection** : check that "Debian desktop" is not checked and that "ssh server" is checked because we do not want a graphical interface (as in the image below)



- **Device for boot loader** : /dev/sda

1.3. Moving the disk image

Right after installation, you need to shut down the virtual machine properly with the following command:

« # systemctl poweroff »

This command must be performed with root, to access root you can use the following command:

« su - » ou « su »

You will be prompted for a password; this is the one you chose during installation (root in this case).

Once the virtual machine is shut down, you can use the following command to upload your virtual machine image to the erebus4 server for easier access later.

« S2.03-déplace-image-disque-sur-erebus4 »

1.4. Debian Server Check

```
root@server-galatn:~# 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=4994253d-6624-4d47-94ec-7bd72bb791d0 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=54a8d2cd-a8ab-4232-962f-61cd95eaa495 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
root@server-galatn:~#
```

Now that your virtual machine image is on the erebus4 server, you can use the virtual machine with the following command:

« S2.03-lance-machine-virtuelle »

By logging in as root by doing the following command:

« cat /etc/fstab » (The result of the command is shown in the image above)

We can see that the virtual machine is installed on /dev/sda1 and that a swap partition is also installed on /dev/sr0 .

If you perform the following command:

« dpkg -l | grep xorg » Then « echo \$? » (To display the result of the first command)

You can see that nothing is returned, this is normal because we do not want a graphical interface (Xorg) for this Debian server.

To verify that your virtual machine can reach outside, you can run the following command which will update apt:

« apt update »

Now I will give you some instructions to access your virtual machine from your Linux workstation (host):

- En SSH, « ssh USER_NAME@localhost -p 2222 »
- En HTTP with l'URL : http://localhost:8080/
- En HTTPS with l'URL : https://localhost:4443/
- With PostgreSQL, « psql -h localhost -U postgres postgres » (Note: The first postgres corresponds to the user and the second postgres corresponds to the database)

2. Apache and PostgreSQL Installation

2.1. Apache Installation

To install Apache, you must first be root and then you must execute the following command:

```
« # apt install apache2 »
```

If there is any problem, here is the link to the Apache installation manual:

<https://httpd.apache.org/docs/2.4/en/install.html>

Execute the command:

```
« # systemctl status apache2 » to verify that apache is started.
```

If not, you can run the following command to start Apache:

```
« # systemctl start apache2 »
```

Without a GUI, you cannot see the HTML page. However, you can connect to the Apache server with the following command:

```
« telnet localhost 80 »
```

A message should appear and then you should write: « HEAD / HTTP/1.0 »

The server should respond with "HTTP/1.1 200 OK" to confirm that Apache is working properly.

You can also access the Apache default homepage on your Linux machine. In the browser, go to the URL: <http://localhost:8080> (as in the image below)



2.2. PostgreSQL Installation

Now you'll install PostgreSQL so you can have your own database on your Debian server.

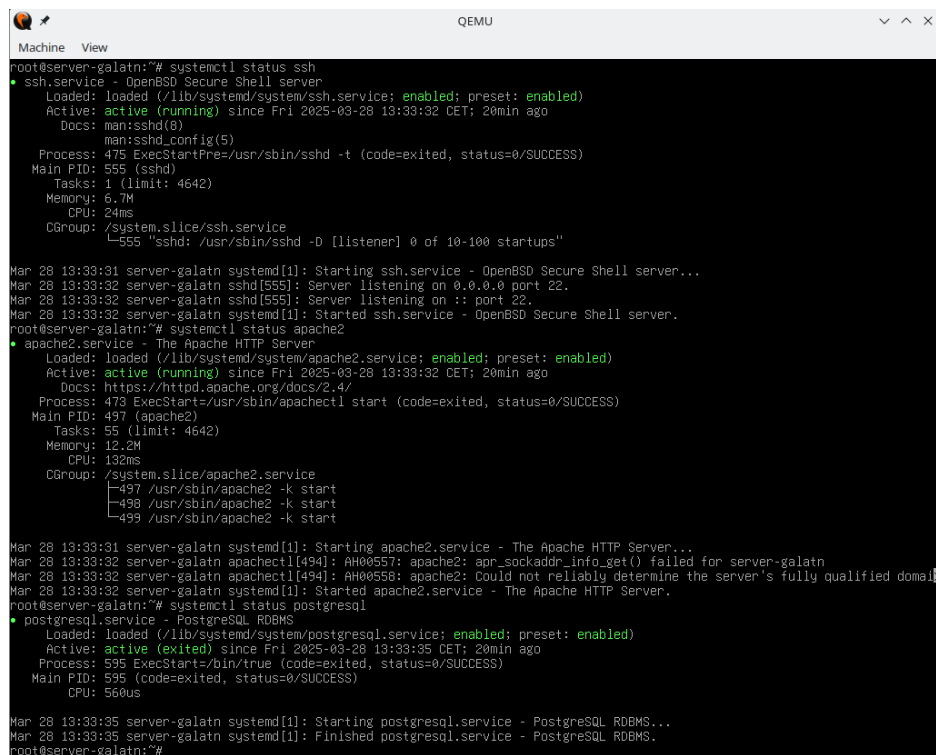
Start by installing the server and client on your virtual machine with the following command:

```
« # apt install postgresql »
```

You can verify the installation with the status command used previously:

```
« systemctl status postgresql »
```

At this point you should have SSH, Apache and PostgreSQL (as shown in the image below)



```
Machine View
root@server-galatn:~# systemctl status ssh
• ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
  Active: active (running) since Fri 2025-03-28 13:33:32 CET; 20min ago
    Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 475 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 555 (sshd)
   Tasks: 1 (limit: 4642)
  Memory: 6.7M
    CPU: 24ms
  CGroup: /system.slice/ssh.service
          └─555 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 28 13:33:31 server-galatn systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Mar 28 13:33:32 server-galatn sshd[555]: Server listening on 0.0.0.0 port 22.
Mar 28 13:33:32 server-galatn sshd[555]: Server listening on :: port 22.
Mar 28 13:33:32 server-galatn systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
root@server-galatn:~# systemctl status apache2
• apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
  Active: active (running) since Fri 2025-03-28 13:33:32 CET; 20min ago
    Docs: https://httpd.apache.org/docs/2.4/
  Process: 473 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 497 (apache2)
   Tasks: 55 (limit: 4642)
  Memory: 12.2M
    CPU: 132ms
  CGroup: /system.slice/apache2.service
          └─497 /usr/sbin/apache2 -k start
            └─498 /usr/sbin/apache2 -k start
              └─499 /usr/sbin/apache2 -k start

Mar 28 13:33:31 server-galatn systemd[1]: Starting apache2.service - The Apache HTTP Server...
Mar 28 13:33:32 server-galatn apachectl[494]: AH00557: apache2: apr_socketaddr_info_get() failed for server-galatn
Mar 28 13:33:32 server-galatn apachectl[494]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, please
Mar 28 13:33:32 server-galatn systemd[1]: Started apache2.service - The Apache HTTP Server.
root@server-galatn:~# systemctl status postgresql
• postgresql.service - PostgreSQL RDBMS
  Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
  Active: active (exited) since Fri 2025-03-28 13:33:35 CET; 20min ago
  Process: 595 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 595 (code=exited, status=0/SUCCESS)
    CPU: 560us

Mar 28 13:33:35 server-galatn systemd[1]: Starting postgresql.service - PostgreSQL RDBMS...
Mar 28 13:33:35 server-galatn systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
root@server-galatn:~#
```

Now that PostgreSQL is installed, you can connect to postgres with the following command:

```
« # su - postgres » then « psql -l » to see the different databases present.
```


2.3. Using PostgreSQL on the virtual machine

Still with postgres, connect to the postgres database with:

« psql » or « psql postgres »

Then create a new user with your name:

« CREATE USER [LOGIN] password 'etu' superuser; »

Then create a base with:

« CREATE DATABASE ma_base with owner = [LOGIN]; »

Remember to enter your login!

You can now connect directly to your database with the user who has your login name.

Now, go to the « my_base » database with the following command:

« \c ma_base »

Then create a simple table:

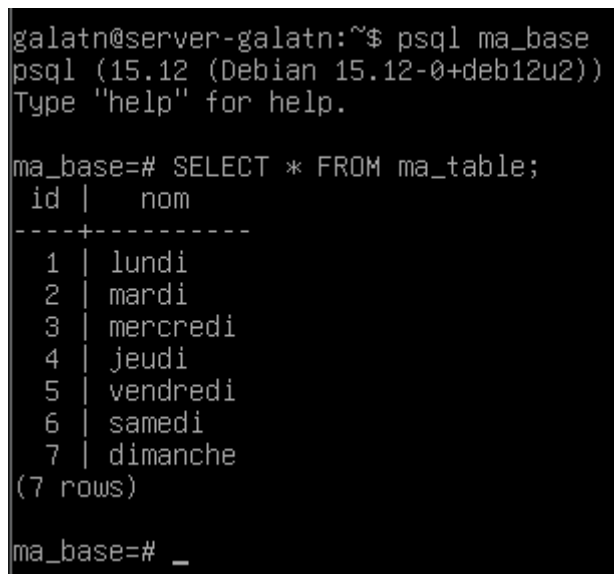
« CREATE TABLE ma_table (id integer, nom varchar); »

Then add a few lines of data to it:

« INSERT INTO ma_table (1,'lundi');

INSERT INTO ma_table (2,'mardi'); ... »

At this point you should be able to query your database with the user who has your login name as in the image below.



```
galatn@server-galatn:~$ psql ma_base
psql (15.12 (Debian 15.12-0+deb12u2))
Type "help" for help.

ma_base=# SELECT * FROM ma_table;
 id |  nom
----+----
  1 | lundi
  2 | mardi
  3 | mercredi
  4 | jeudi
  5 | vendredi
  6 | samedi
  7 | dimanche
(7 rows)

ma_base=# _
```

2.4. Using PostgreSQL on Linux Machine

To connect directly to your database on your Linux machine, you need to modify a few files to provide access to it.

Start by editing the first file « postgresql.conf » with the command:

```
« nano /etc/postgresql/15/main/postgresql.conf »
```

Look in the CONNECTIONS AND AUTHENTICATION section for the following line to uncomment and update:

```
« listen_addresses = '*' »
```

Reminder of basic nano commands:

Ctrl-w : research

Ctrl-o : save

Ctrl-x : exit

After editing this file, you can edit the second file « pg_hba.conf » with the following command:

```
« nano /etc/postgresql/15/main/pg_hba.conf »
```

Change the « IPv4 local connections » part to :

```
« host all all 0.0.0.0/0 scram-sha-256 »
```

Now restart the postgresql server with:

```
« service postgresql restart »
```

With the command « psql -h localhost my_base » on your Linux machine, you should be able to connect to your database and query it like the image below.

```
galatn@iut2-dg037-d13:~$ psql -h localhost ma_base
Password for user galatn:
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.

ma_base=# SELECT * FROM ma_table;
 id |  nom
----+-----
  1 | lundi
  2 | mardi
  3 | mercredi
  4 | jeudi
  5 | vendredi
  6 | samedi
  7 | dimanche
(7 rows)

ma_base=#
```

2.5. Database verification

```
galatn@server-galatn:~$ psql -l
```

Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider	Access privileges
ma_base	galatn	UTF8	en_US.UTF-8	en_US.UTF-8		libc	
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 +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8		libc	=c/postgres +

```
(4 rows)
galatn@server-galatn:~$ _
```

Now you should have the databases as shown in the image above. With the owner of your database (here my_base) being the user with your login name.

You can view the password hashes in the "pg_shadow" table.
To do this, simply perform a Select from the "pg_shadow" table.

« SELECT * FROM pg_shadow; »

You should have a result similar to the image below.

```
ma_base=# SELECT * from pg_shadow;
```

username	usesysid	usecreatedb	usesuper	use repl	usebypassrls	passwd	valuntil	useconfig
postgres	10	t	t	t	t			
galatn	16388	f	t	f	f	SCRAM-SHA-256\$4896:ckz8cf1xv97dlv0k+MBF2Q==3UG9C46fbKnyvEd4YgMpLLYw75LwPq1rQ0130hptfE=:ZK081PFMKpncPm1CF/1mG6EhXFPUG0G6Vjt6aQu90=		

```
(2 rows)
ma_base=#
```

The postgres user is a superuser with full privileges, including database creation, replication, and the ability to bypass security policies (RLS).

On the other hand, your user has the right to create databases but does not have administration or replication privileges. Their password is stored encrypted using the secure SCRAM-SHA-256 algorithm.

This information is sensitive and reserved for PostgreSQL superusers.

3. PHP and PhpPgAdmin installation

3.1. PHP Installation

To install PHP, you must first be root and then you must execute the following command:

```
« # apt install php-common libapache2-mod-php php-cli »
```

If there is any problem, here is the link to the PHP installation manual:

<https://www.php.net/manual/en/install.unix.php>

From here you can put PHP files on your server and access them.

The directory that allows you to access your PHP files locally: « /var/www/html/ »

You will try it with the file "page_sae_S2.03.php" that you will copy from transit.

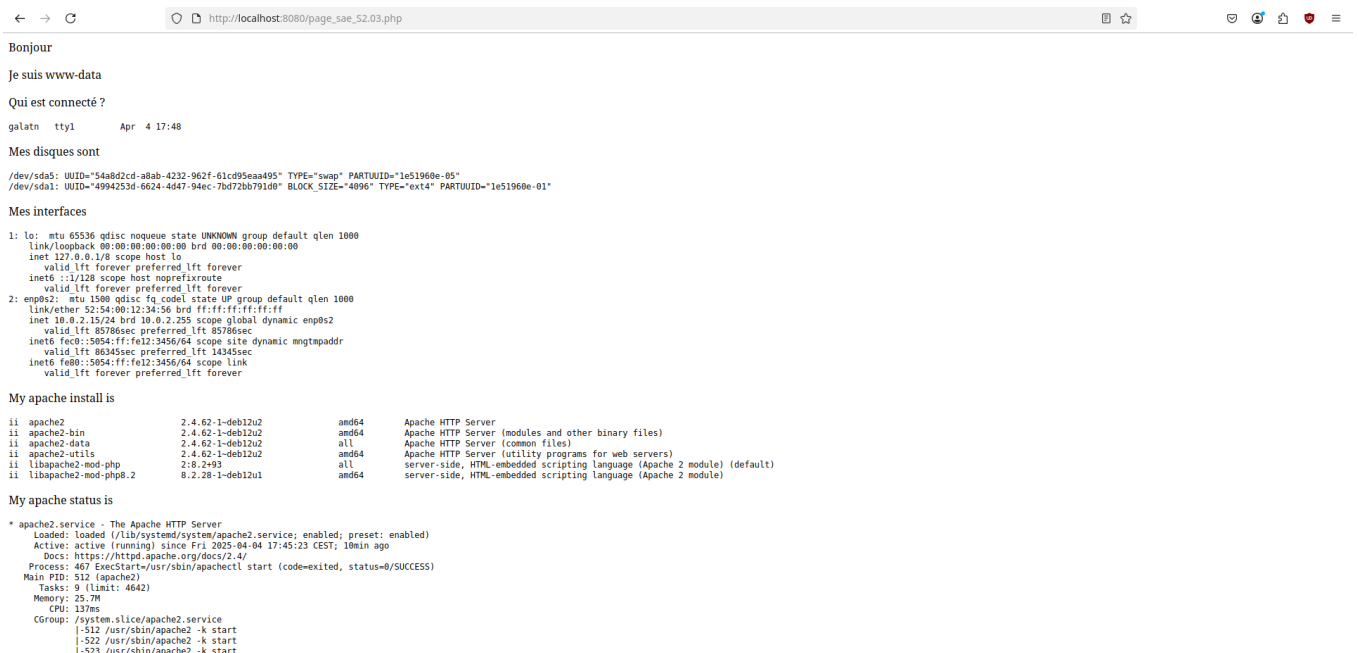
To do this, use the following command:

```
« scp LOGIN@transit.iut2.univ-grenoble-alpes.fr:/users/info/www/intranet/enseignements/S2.03/
page_sae_S2.03.php /var/www/html/ »
```

You must type yes and then enter your transit password.

If you go to the following link: http://localhost:8080/page_sae_S2.03.php

You should land on the page like the image below.



```
Bonjour

Je suis www-data

Qui est connecté ?

galatin tyti Apr 4 17:48

Mes disques sont

/dev/sda5: UUID="54a8d2cd-a8ab-4232-962f-61cd95eaa495" TYPE="swap" PARTUUID="1e51960e-05"
/dev/sda1: UUID="4994253d-6624-4647-94ec-7bd72bb791d0" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="1e51960e-01"

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 85786sec preferred_lft 85786sec
    inet6 fec8::5054:ff:fe12:3456/64 scope site dynamic mngtmpaddr
        valid_lft 86345sec preferred_lft 14345sec
    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 (modules and other binary files)
ii apache2-bin 2.4.62-1-deb12u2 amd64 Apache HTTP Server (common files)
ii apache2-data 2.4.62-1-deb12u2 all Apache HTTP Server (utility programs for web servers)
ii apache2-utils 2.4.62-1-deb12u2 amd64 server-side, HTML-embedded scripting language (Apache 2 module) (default)
ii libapache2-mod-php 2:0.2+93 all server-side, HTML-embedded scripting language (Apache 2 module)
ii libapache2-mod-php8.2 8.2.28-1-deb12u1 amd64

My apache status is

* apache2.service - The Apache HTTP Server
Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
Active: active (running) since Fri 2025-04-04 17:45:23 CEST; 10min ago
Docs: https://httpd.apache.org/docs/2.4/
Process: 407 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
Main PID: 512 (apache2)
Tasks: 9 (limit: 4642)
Memory: 25.7M
CPU: 137ms
CGroup: /system.slice/apache2.service
        └─512 /usr/sbin/apache2 -k start
           └─522 /usr/sbin/apache2 -k start
             └─523 /usr/sbin/apache2 -k start
```

3.2. PhpPgAdmin Installation

To install PhpPgAdmin, you need the package located at:

<http://ftp.de.debian.org/debian>

To access this package, you need to modify the apt sources present in the file « sources.list »

So you will use the following command:

« nano /etc/apt/sources.list »

Then add the following line: « deb http://ftp.de.debian.org/debian bookworm-backports main »

When this is done, save (Ctrl-s) and exit (Ctrl-x) then update apt with:

« apt update »

Now you can download PhpPgAdmin with the following command:

« # apt install phppgadmin »

Now we will have to enable the connection by modifying the file « Connection.php »

So run the command: « nano /usr/share/phppgadmin/classes/database/Connection.php »

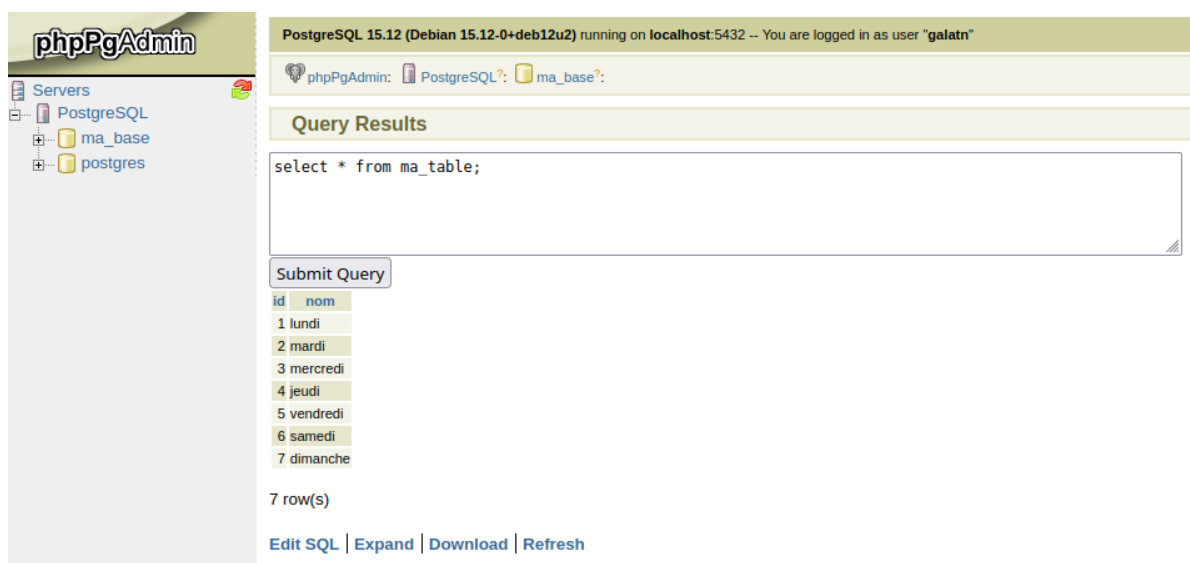
And modify the line: « case '14': return 'Postgres'; break; »

by « case '15': return 'Postgres'; break; »

You can use PhpPgAdmin by going to the following link:

<http://localhost:8080/phppgadmin/>

Then you can query your database as in the image below.



4. Security

At this point, you should have a storage space similar to the image below.

Try the command: « `df -h` »

```
root@server-galatn:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G   0    1.9G   0% /dev
tmpfs           392M  480K  392M   1% /run
/dev/sda1       3.0G  1.6G  1.3G  56% /
tmpfs           2.0G  1.1M  2.0G   1% /dev/shm
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           392M   0    392M   0% /run/user/1000
root@server-galatn:~#
```

However, security vulnerabilities are regularly discovered in Debian packages, the solution is to regularly update the system and its packages with:

« `# apt update && apt upgrade -y` »

You can also remove unnecessary software such as:

« `# systemctl disable bluetooth` »

« `# systemctl disable avahi-daemon` »

To strengthen your system security, you can download software that will prevent brute force attacks, such as Fail2ban.

« `# apt install fail2ban` »

« `# systemctl enable --now fail2ban` »

You can also set up a firewall with UFW:

« `# ufw enable` »

« `# ufw allow OpenSSH` »

« `# ufw allow 'Apache Full'` »

As for Apache, you can start by hiding the version by editing the « `security.conf` » file with the following command:

« `nano /etc/apache2/conf-available/security.conf` » and putting the following lines :

`ServerTokens Prod`

`ServerSignature Off`

You can also use HTTPS with Let's Encrypt for example:

« `# apt install certbot python3-certbot-apache` »

« `# certbot --apache` »

All these measures will help you better secure your Debian server.

5. Command Summary

COMMANDS :	DESCRIPTION
sha512sum [FICHER]	Calculates the SHA-512 hash of the file to verify its integrity
S2.03-lance-installation	Starts the Debian installation procedure
S2.03-déplace-image-disque-sur-erebus4	Move the disk image to the "erebus4" machine
S2.03-lance-machine-virtuelle	Launch the virtual machine
# systemctl poweroff	Cleanly shuts down the virtual machine
su -	Switch to root user
cat [CHEMIN]	Displays the contents of a file
apt update	Updates the list of available packages.
ssh USER_NAME@localhost -p 2222	Connects via SSH to your own machine via port 2222
psql -h localhost -U [USER] [BD]	Connects to a local PostgreSQL database with the given user.
# apt install [LOGICIEL]	Install software with apt
# systemctl status [LOGICIEL]	Displays the status of a software
# systemctl start [LOGICIEL]	Starts a software
# su - postgres	Switch to the postgres system user
psql -l	List all PostgreSQL databases
psql [USER]	Opens a database with the chosen user
service postgresql restart	Restarts the PostgreSQL service.
nano [CHEMIN]	Open a file with the Nano editor
scp [CHEMIN_SOURCE] [CHEMIN DEST]	Copy a file remotely via SSH
df -h	Shows available and used space.
SQL :	-----
CREATE USER [LOGIN] [OPTION];	Create a new PostgreSQL user
CREATE DATABASE [BASE] WITH owner = [USER];	Creates a new database named and adds an owner to it
SELECT * FROM pg_shadow;	Displays the pg_shadow system table
\c [BD]	Changing databases in PostgreSQL

6. Installing a virtual machine on a USB key (annex)

To create a virtual machine on a USB stick, you must first prepare the USB stick by formatting it in exFAT (or ext4) so that you can create the disk without any problems.

Once the key is formatted, you need to create your Debian server's virtual hard drive directly on your USB drive. You can use the following command (here, 20 GB is allocated):

```
« qemu-img create -f qcow2 /media/$USER/KEY_NAME/debian-vm.qcow2 20G »
```

Once this command is complete, you need to download and place the Debian ISO on the drive in an iso directory (iso/debian-12.10.0-amd64-netinst.iso).

Here, you can also double-check the ISO with SHA (see 1.1).

Next, you need to reuse the "S2.03-commun" script, modifying and adding a few things:

```
qemu-system-x86_64 \  
-machine q35 \  
-cpu host \  
-m 4G \  
-enable-kvm \  
-device VGA,xres=1024,yres=768 \  
-display gtk,zoom-to-fit=off \  
-cdrom /media/usb/iso/debian-12.iso \                               (added)  
-boot d \                                                            (added)  
-drive file=/media/usb/debian-vm.qcow2,format=qcow2 \               (modified)  
-device e1000,netdev=net0 \  
-netdev user, id=net0,hostfwd=tcp::2222-:22, hostfwd=tcp::4443-:443, hostfwd=tcp::8080-:80,  
hostfwd=tcp::5432-:5432
```

Your server will install on your USB drive (be sure to check the paths carefully!)

You can put this script (without the cdrom and boot lines) in a .sh file (e.g. start_vm.sh) so you don't have to re-run the script at each startup and simply execute:

```
« # ./start_vm.sh »
```

The installation of the different software remains similar, however, because the Debian virtual machine works in the same way, even if it is hosted on a USB key.