# SAE 2.03
# Installation guide

By
Leane Buisson

# Contents

In this installation guide we will try to install a virtual machine on Debian with the Qemu/KVM. In this guide you will learn how to install and format your own virtual machine. We will also see how to add some external functionality like PostgreSQL, PHP and PhpPgAdmin.

# Part 1: Debian Installation

## 1. Prepare Installation

**Step 1 →** Download the ISO file with:

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

**Step 2 →** Check your ISO picture repertory's contents and be sure you have all that you have downloaded:

```
~$ ls -l /usr/local/images-ISO/
```

**Step 3 →** Check the entirety of you ISO picture and compare the 2 mark

Within the scope of the SAE project, I'll use Qemu/KVM as software.

**Step 4 →** As part of the SAE project, the start instruction is:

```
$ S2.03-lance-installation
```

This command is written in the Qemu's line command process, which allows it to run the Qemu's interface.
To launch your Qemu just run the command: `$ lance_qemu`
→ which is contained in your virtual machine creation file.

```
lance_qemu="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 $drive -device
e1000,netdev=net0 -netdev
user,id=net0,hostfwd=tcp::2222-:22,hostfwd=tcp::4443-:443,hostfwd=tcp::8080-:80,h
ostfwd=tcp::5432-:5432"
```

- `qemu-system-x86_64`

  → this is the main QEMU binary.

- `-machine q35`

  → the virtual machine model.

- `-cpu host`

  → Qemu use host CPU for emulation

❖ `-m 4G`

→ allocates 4 GB of memory to the virtual machine.

❖ `-enable-kvm`

→ support for hardware virtualization (KVM). QEMU can use CPU virtualization extensions to boost performance.

❖ `-device VGA,xres=1024,yres=768`

→ allows the display of the virtual machine. xres and yres represent a resolution of 1024x768 pixels.

❖ `-display gtk,zoom-to-fit=off`

→ use the GTK interface (GUI). The option zoom-to-fit=off disables automatic zooming of the GUI.

❖ `-drive $drive`

→ path to disk image. $ drive should be replaced with the real path to the disk image.

❖ `-device e1000,netdev=net0`

→ virtual network adapter of type e1000 to the virtual machine. It also links to the network device net0.

❖ `-netdev user,id=net0,hostfwd=tcp::2222-:22,hostfwd=tcp::4443-:443,hostfwd=tcp::8080-:80,hostfwd=tcp::5432-:5432`

→ this creates a virtual network device of type user with the identifier net0 and allows connection to the internet. The hostfwd options define the forwarding of TCP ports from the host to the virtual machine.

## 2. System Installation

Now following the next stages. When nothing is specified, choose the default parameter.

**Step 1 →** Choose the basic language (as you want): English

**Step 2 →** Select your location: other → Europe → England (for example)

**Step 3 →** Choose the locales (basic parameter): United States, en_US.UTF-8

**Step 4 →** Take a keyboard that correspond to yours: English (for example)

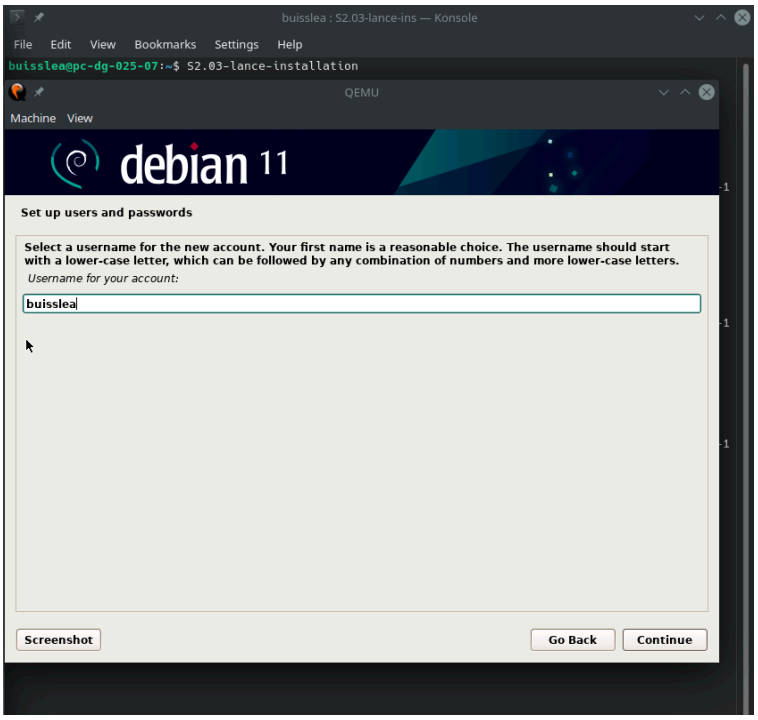
**Step 5 →** Create your hostname: use server-"name that you want"

**Step 6 →** Domain name is not really important.

**Step 7 →** Add a root password, choose an easy password that you can remember. For example you can choose "root", a simple password will not be problematic for security.

**Step 8 →** Write your user account name: Full name (ex: "John Smith")

**Step 9 →** Write your user name: that will be your login




**Step 10 →** Choose your user password: that will be your login password

**Step 11 →** Partition disks: Guided - use entire disk

**Step 12 →** Partition disks: All files in one partition

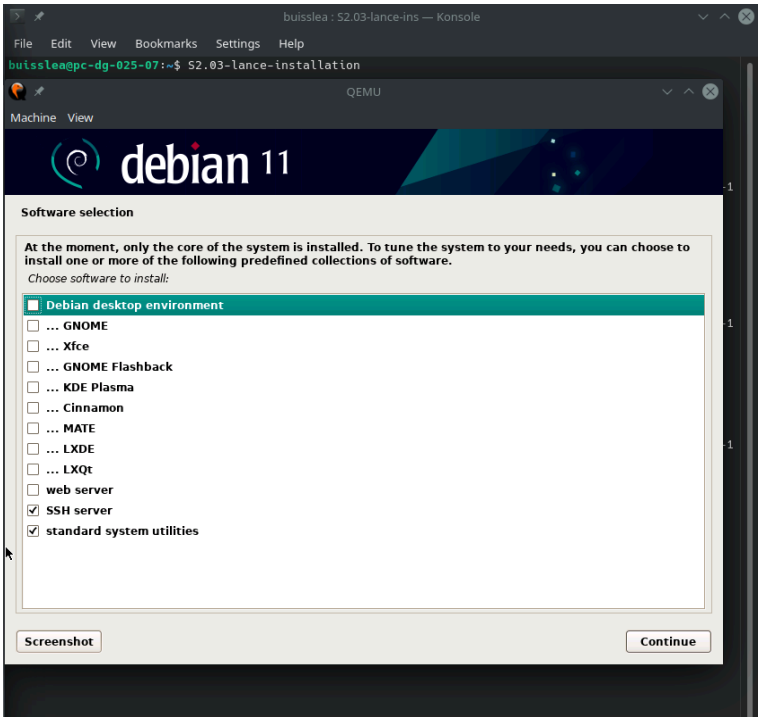
**Step 13 →** Partition disks: YES (that's very important)

**Step 14 →** Scan media: No

**Step 15 →** Select the country language that you prefer: England

**Step 16 →** Select the good debian archive: deb.debian.org

**Step 17 →** Software Selection: make you sure that **“Debian desktop” is uncheck** and that you **check “ssh server”**




**Step 18 →** Install GRUB: Yes

**Step 19 →** Choose the device for boot loader: **/dev/sda** (advise this one)

==> Connect you to superuser, in root with: `$ su -`
Switch off your virtual machine with the command: *poweroff* **(don’t close brutally the window)**

# Part 2: Check and Connection

## 1. Check Debian servers

- Check your Ethernet and IP: *$ ip addr*

and also check the outside connection with pinging another machine with the command: *$ ping [+adresse]*

- Make sure your don't have Xorg server: *$ dpkg -l | grep xorg*

X is a windowing system protocol that manages the screen, the mouse and also the keyboard. It is the open standard for graphical user interaction on Unix-like operating systems.

=> our machine is not equipped with it because we don't want a graphical interface.

## 2. Try to connect to ssh

Connecting to ssh is recommended to have better use of your virtual machine and to be able to access it from anywhere.

- Check if SSH is started:
  *# systemctl status ssh*

```
root@server-buisslea:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-05-02 14:37:45 CEST; 39min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 422 (sshd)
      Tasks: 1 (limit: 4661)
     Memory: 5.4M
        CPU: 67ms
     CGroup: /system.slice/ssh.service
             └─422 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

May 02 14:37:45 server-buisslea systemd[1]: Starting OpenBSD Secure Shell server...
May 02 14:37:45 server-buisslea sshd[422]: Server listening on 0.0.0.0 port 22.
May 02 14:37:45 server-buisslea sshd[422]: Server listening on :: port 22.
May 02 14:37:45 server-buisslea systemd[1]: Started OpenBSD Secure Shell server.
May 02 14:59:45 server-buisslea sshd[483]: Accepted password for buisslea from 10.0.2.2 port 47240 s>
May 02 14:59:45 server-buisslea sshd[483]: pam_unix(sshd:session): session opened for user buisslea(>
lines 1-18/18 (END)
```

- Use the command: `$ ssh [your login]@localhost -p 2222`
  (2222 is the session port were you are trying to connect you to the ssh)

- Connect yourself to the root with: `$ su -`
  (you can log off with: `# exit`)

- For being sure that it's ok, you can download something like a text editor, "micro" for example:
  - write the command: `$ apt search micro` (for check that micro is on the apt)
  - install the text editor with: `$ apt install micro`
  - write a new text file as test

# Part 3: Apache Installation

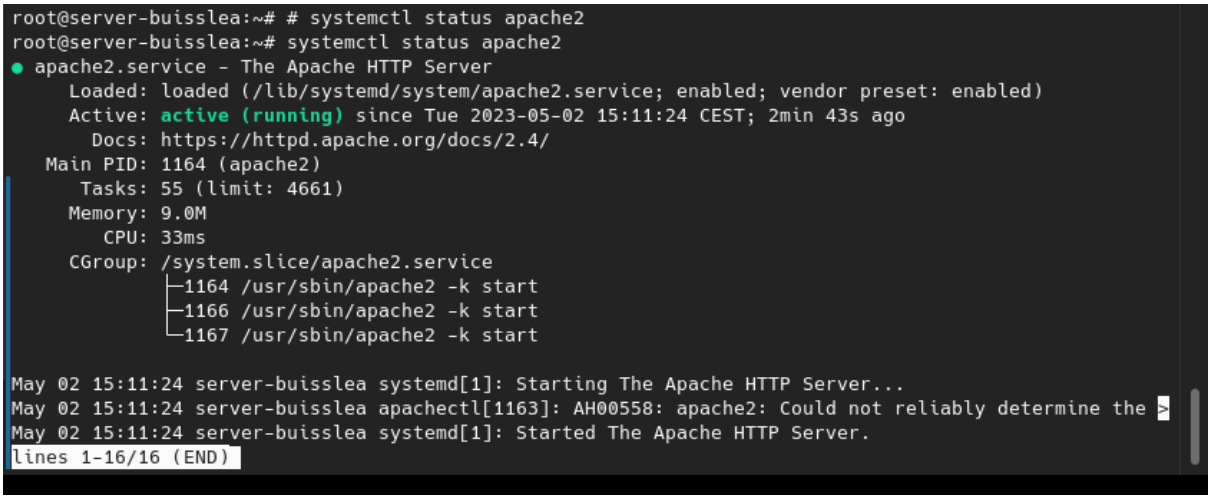
## 1. Download Apache

**Step 1 →** Connect yourself to the root with: *$ su -*

**Step 2 →** Install Apache with the command: *apt install apache2*

**Step 3 →** Check if Apache is Load:

*# systemctl status apache2*




If Apache isn't running, restart the server with: *# systemctl restart apache2*

## 2. Configure Apache

**Step 1 →** Connect yourself to Apache with: *telnet localhost 80*

**Step 2 →** Send to the server: *HEAD / HTTP/1.0 (add 2 lines break after to confirm the message)*

=> The server should respond: *HTTP/1.1 200 OK*

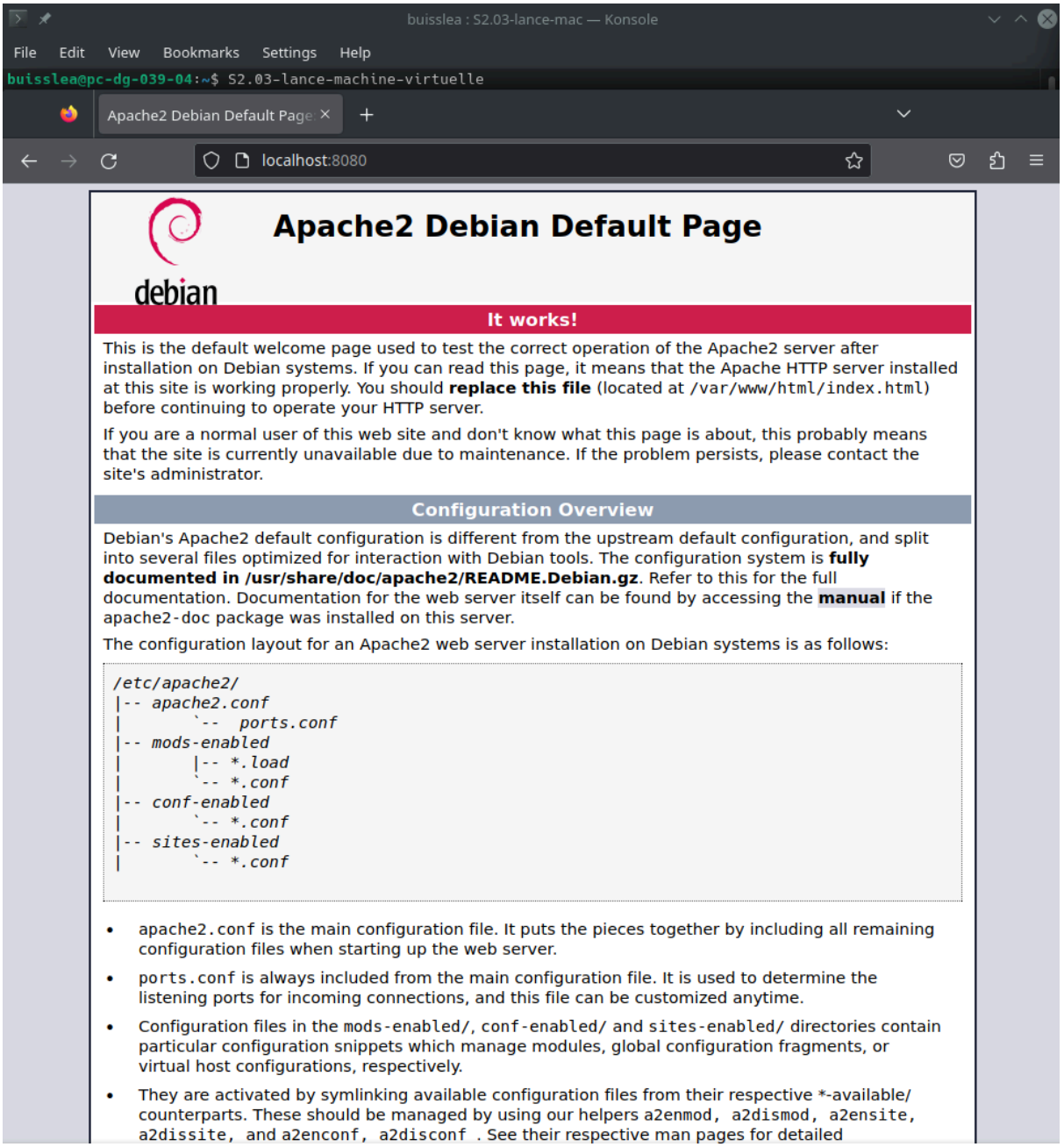
For example:

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

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

Step 3 → You can check your Apache server in a browser on your host machine with this URL:

http://localhost:8080




On this web page you have all the information of your Apache server, you can check it when you want or when you need.

# Part 4: PostgreSQL Installation

## 1. Download PostgreSQL

**Step 1 →** Connect yourself to the root with: `su -`

**Step 2 →** Install PostgreSQL with the command: `apt install postgresql`

**Step 3 →** Check if PostgreSQl is Load:

`# systemctl status postgresql`

```
root@server-buisslea:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Wed 2023-05-10 14:57:08 CEST; 12min ago
    Process: 3784 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 3784 (code=exited, status=0/SUCCESS)
        CPU: 824us

May 10 14:57:08 server-buisslea systemd[1]: Starting PostgreSQL RDBMS...
May 10 14:57:08 server-buisslea systemd[1]: Finished PostgreSQL RDBMS.
root@server-buisslea:~#
```

- You can check with the command too: `# pg_lsclusters`
- You can also try a connection with the postgres login to check that it is working correctly : `# su - postgres`

**Step 4 →** You can access to the default database with: `$ psql -l`

**Step 5 →** Connect yourself at your local PostgreSQL server: `$ psql`
(you can log off with: `\q` )

## 2. Configure PostgreSQL

In a postgres connection:

**Step 1 →** Create your user:

```
CREATE USER yourLogin WITH password 'yourPassword';
```

**Step 2 →** Alter your user to give some access to the database, here we just need to manage the base:

```
ALTER USER buisslea WITH createdb;
```

→ “buisslea” is my login but you put yours

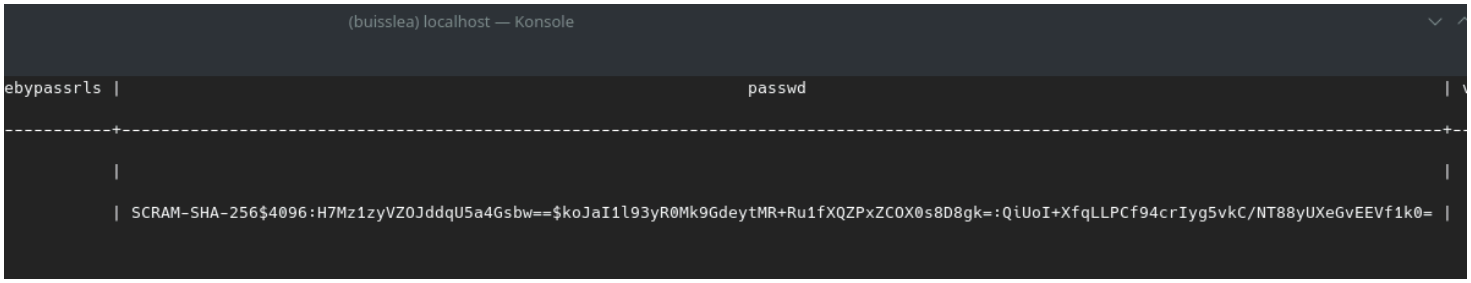
Now we will try to alter some parameters to allow you to connect to an external machine and to allow you to create your database easily.

- Edit the configuration file:
  - access to this file with:
    *# micro /etc/postgresql/13/main/postgresql.conf*
  - alter the “listen_adresses“ line by removing the comment mode (#) and replace ‘localhost’ with ‘*’
  - remove also the comment mode of the “password_encryption = scram-sha-256“ line ( and replace “md5“ by “scram-sha-256“ if you need)

- Edit the authentication rules file:
  - access to this file:
    *# micro /etc/postgresql/13/main/pg_hba.conf*
  - add this line (where you want in the file):
    *#IPv4 remote connections:*
    *host all all 0.0.0.0/0 scram-sha-256*
  - change also all the occurrences ‘md5’ by ‘scram-sha-256'

- Restart the PostgreSQL server:
  *service postgresql restart*

- Check the password type:
  → connect yourself to psql: *$ psql*
  → execute the command:
  *SELECT * FROM pg_shadow;*




=> it will return a table with a column passwd where you can see your password encryption

Try to connect yourself in a external shell (login on your server) with the command:
yourLogin@server-yourLogin: ~$ *psql -h localhost postgres*

## 3. Fill your base up

If you are not connected to postgres, take the command above.

**Step 1 →** Create a new database:

*CREATE DATABASE ma_base;*

→ that will create a base with “ma_base” as name (you can change it)

**Step 2 →** You can check the creation with: *\l*

```
buisslea@ma_base=> \l
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 ma_base   | buisslea | 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)

buisslea@ma_base=>
```

→ As you can see your new database is on the top of the list, you can manage the access privileges with SQL commands.

**Step 3 →** Now you can connect yourself to your base:

*\c ma_base yourLogin* (*yourLogin* created on the create user step)

**Step 4 →** Then create a new table:

```
CREATE TABLE MA_TABLE(
col1 varchar primary key,
col2 numeric(3),
col3 varchar NOT NULL
);
```

→ that will create a table with “ma_table” as name (you can change it)
→ the several parameters can be change too

**Step 5 →** Insert contents in your table:

```
INSERT INTO MA_TABLE values ('testUn', 38, 'blblblblbl');
INSERT INTO MA_TABLE values ('testDeux', 39,'nmnmnmnmnmnmnm');
INSERT INTO MA_TABLE values ('testTrois', 40,'vvvvvvvvvvvvv');
```

/!\ be careful of the type of each column

**Step 6 →** You can check the creation with: *\d*

```
buisslea@ma_base=> \d
         List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | ma_table | table | buisslea
(1 row)
```

**Step 7 →** To test the previous commands you can execute a select command:

- on your virtual machine:

  *SELECT * FROM ma_table;*

```
buisslea@server-buisslea:~$ psql ma_base -U buisslea
psql (13.11 (Debian 13.11-0+deb11u1))
Type "help" for help.

ma_base=> SELECT * FROM ma_table;
   col1    | col2 |      col3
-----------+------+----------------
 testUn    |   38 | blblblblbl
 testDeux  |   39 | nmnmnmnmnmnmnm
 testTrois |   40 | vvvvvvvvvvvvv
(3 rows)

ma_base=>
```

- on an external connexion:

  → connect yourself to:

  *$ psql -h localhost ma_base -U yourLogin*

  → write your select command:

  *SELECT * FROM ma_table;*

```
buisslea@pc-dg-035-12:~$ psql -h localhost ma_base -U buisslea
Password for user buisslea:
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.

buisslea@ma_base=> \d
          List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | ma_table | table | buisslea
(1 row)

buisslea@ma_base=> SELECT * FROM ma_table;
   col1    | col2 |      col3
-----------+------+----------------
 testUn    |   38 | blblblblbl
 testDeux  |   39 | nmnmnmnmnmnmnm
 testTrois |   40 | vvvvvvvvvvvvv
(3 rows)
```

If you don't have any result, you can *DROP* each database and table and try again to fill in your base .

# Part 5: PHP Installation

**Step 1 →** Connect yourself to the root with: *$ su -*

**Step 2 →** Install PHP with the command:

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

**Step 3 →** Restart the device with:

- *# /etc/init.d/apache2 stop*
- *# /etc/init.d/apache2 start*

**Step 4 →** Create a info.php file in the folder: /var/www/html
for this I recommend you to do:

```
$ micro /var/www/html/info.php
```

(if the file doesn't exist *micro* will create a new file in the correct directory)

**Step 5 →** In this file write those instructions:

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

**Step 6 →** Execute this command on your virtual machine: */sbin/blkid*

**Step 7 →** Then if you want a PHP file with all server information:

- create new file with *.php* at the end
- or copy an existing file in your virtual machine with a *scp* command like:

```
scp
buisslea@transit.iut2.univ-grenoble-alpes.fr:/users/info/www/intrane
t/enseignements/S2.03/page_sae_S2.03.php /var/www/html/
```

with:

*scp* : copy command
*buisslea@transit.iut2.univ-grenoble-alpes.fr* : the host's adresse
*/users/info/www/intranet/enseignements/S2.03/page_sae_S2.03.php* : the location on the host
*/var/www/html/* : the location where you wanna copy your file

**Step 8 →** You can now open your PHP file on a browser:
http://localhost:8080/page_sae_S2.03.php

# Your PHP file look like this one:

Bonjour

Je suis www-data

Qui est connecté ?

```
buisslea pts/0        Jun  6 17:00 (10.0.2.2)
```

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
       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 86337sec preferred_lft 86337sec
    inet6 fec0::5054:ff:fe12:3456/64 scope site dynamic mngtmpaddr
       valid_lft 86337sec preferred_lft 14337sec
    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) (default)
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 Tue 2023-06-06 17:00:03 CEST; 1min 4s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 410 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 464 (apache2)
      Tasks: 8 (limit: 4661)
     Memory: 22.5M
        CPU: 127ms
     CGroup: /system.slice/apache2.service
             |-464 /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
             |-582 sh -c systemctl status apache2
             `-583 systemctl status apache2
```

My postgresql install is

```
ii  postgresql                   13+225                        all          object-relational SQL database (supported version)
ii  postgresql-13                13.11-0+deb11u1               amd64        The World's Most Advanced Open Source Relational Database
ii  postgresql-client-13         13.11-0+deb11u1               amd64        front-end programs for PostgreSQL 13
ii  postgresql-client-common     225                           all          manager for multiple PostgreSQL client versions
ii  postgresql-common            225                           all          PostgreSQL database-cluster manager
```

My postgresql status is

```
* postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2023-06-06 17:00:05 CEST; 1min 2s ago
    Process: 542 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 542 (code=exited, status=0/SUCCESS)
        CPU: 956us
```

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 Tue 2023-06-06 17:00:03 CEST; 1min 4s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 412 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 452 (sshd)
      Tasks: 1 (limit: 4661)
     Memory: 6.1M
        CPU: 109ms
     CGroup: /system.slice/ssh.service
             `-452 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
```

# Part 6: PhpPgAdmin Installation

## 1. Download PhpPgAdmin

**Step 1 →** Find PhpPgAdmin in your virtual machine library:

```
# apt search phppgadmin
```

**Step 2 →** Install PhpPgAdmin in your virtual machine:

```
# apt install phppgadmin
```

## 2. Access to the web interface

**Step 1 →** Restart the device:

- `/etc/init.d/apache2 stop`
- `/etc/init.d/apache2 start`

**Step 2 →** Open a web page and search the following URL:
http://localhost:8080/phppgadmin

## 3. PhpPgAdmin guide

To access your databases or tables on PhpPgAdmin, following the next stages.

**Step 1 →** Connect yourself on the postgres server: click on SERVER

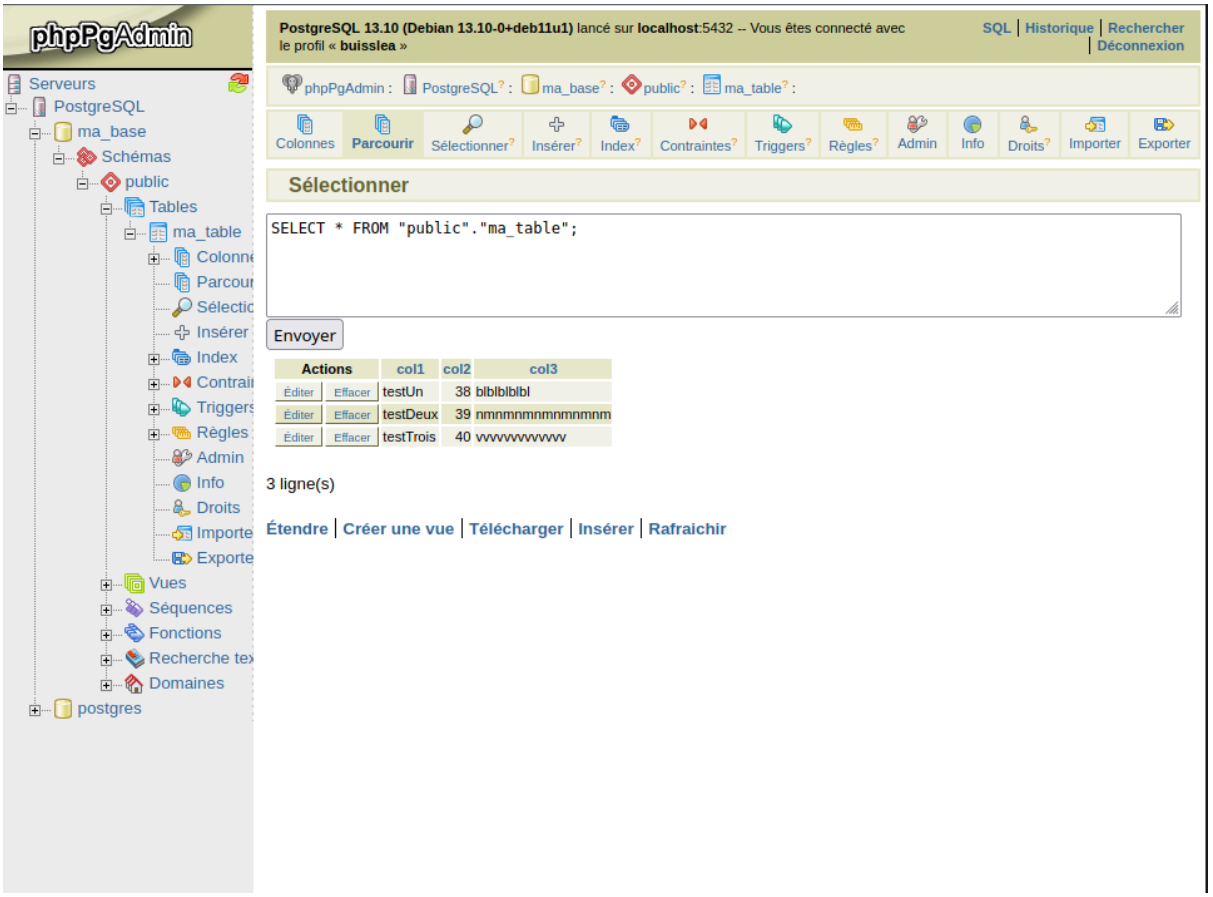
**Step 2 →** Access to your base: open the BASE drop-down menu ‘click on the + )

**Step 3 →** Access to your table: open the TABLE drop-down menu ‘click on the + )

**Step 4 →** Choose your table by clicking on its NAME

**Step 5 →** Choose the interaction type like a SELECT for example (here, SELECT = Parcourir)

**Step 6 →** In the text area write your SQL query, click on “send” (or “envoyer” in french) to get a result




This interface is a french version but you can find the representative elements of the SQL language.

Below the text area, there is a table containing the results of the query.

# Part 7: Check System

- First, check the storage space with: `# df -h`
  → be sure that your storage space has not been impacted too much.

```
root@server-buisslea:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           392M  500K  392M   1% /run
/dev/sda1       3.0G  1.4G  1.5G  49% /
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
root@server-buisslea:~#
```

- Next, we will fix, the possible, security vulnerabilities of your system:

  **Step 1 →** Search potential updates: `# apt update`

```
root@server-buisslea:~# apt update
Hit:1 http://security.debian.org/debian-security bullseye-security InRelease
Hit:2 http://deb.debian.org/debian bullseye InRelease
Hit:3 http://deb.debian.org/debian bullseye-updates InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

**Step 2 →** Update the different packages: `# apt upgrade`

(you can see an example on the next page)

```
root@server-buisslea:~# apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  linux-image-5.10.0-23-amd64
The following packages will be upgraded:
  libpq5 linux-image-amd64 postgresql-13 postgresql-client-13
4 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 72.4 MB of archives.
After this operation, 318 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.debian.org/debian-security bullseye-security/main amd64 libpq5 amd64 13.11-0+de
b11u1 [180 kB]
Get:2 http://security.debian.org/debian-security bullseye-security/main amd64 linux-image-5.10.0-23-a
md64 amd64 5.10.179-1 [55.6 MB]
Get:3 http://security.debian.org/debian-security bullseye-security/main amd64 linux-image-amd64 amd64
 5.10.179-1 [1,484 B]
Get:4 http://security.debian.org/debian-security bullseye-security/main amd64 postgresql-client-13 am
d64 13.11-0+deb11u1 [1,512 kB]
Get:5 http://security.debian.org/debian-security bullseye-security/main amd64 postgresql-13 amd64 13.
11-0+deb11u1 [15.1 MB]
Fetched 72.4 MB in 4s (17.3 MB/s)
Reading changelogs... Done
Preconfiguring packages ...
(Reading database ... 31731 files and directories currently installed.)
Preparing to unpack .../libpq5_13.11-0+deb11u1_amd64.deb ...
Unpacking libpq5:amd64 (13.11-0+deb11u1) over (13.10-0+deb11u1) ...
Selecting previously unselected package linux-image-5.10.0-23-amd64.
Preparing to unpack .../linux-image-5.10.0-23-amd64_5.10.179-1_amd64.deb ...
Unpacking linux-image-5.10.0-23-amd64 (5.10.179-1) ...
Preparing to unpack .../linux-image-amd64_5.10.179-1_amd64.deb ...
Unpacking linux-image-amd64 (5.10.179-1) over (5.10.178-3) ...
Preparing to unpack .../postgresql-client-13_13.11-0+deb11u1_amd64.deb ...
Unpacking postgresql-client-13 (13.11-0+deb11u1) over (13.10-0+deb11u1) ...
Preparing to unpack .../postgresql-13_13.11-0+deb11u1_amd64.deb ...
Unpacking postgresql-13 (13.11-0+deb11u1) over (13.10-0+deb11u1) ...
Setting up linux-image-5.10.0-23-amd64 (5.10.179-1) ...
I: /vmlinuz is now a symlink to boot/vmlinuz-5.10.0-23-amd64
I: /initrd.img is now a symlink to boot/initrd.img-5.10.0-23-amd64
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-5.10.0-23-amd64
/etc/kernel/postinst.d/zz-update-grub:
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.10.0-23-amd64
Found initrd image: /boot/initrd.img-5.10.0-23-amd64
Found linux image: /boot/vmlinuz-5.10.0-22-amd64
Found initrd image: /boot/initrd.img-5.10.0-22-amd64
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
done
Setting up libpq5:amd64 (13.11-0+deb11u1) ...
Setting up linux-image-amd64 (5.10.179-1) ...
Setting up postgresql-client-13 (13.11-0+deb11u1) ...
Setting up postgresql-13 (13.11-0+deb11u1) ...
Processing triggers for postgresql-common (225) ...
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Removing obsolete dictionary files:
Processing triggers for libc-bin (2.31-13+deb11u6) ...
root@server-buisslea:~#
```

→ we can see that the postgresql package has been updated so its security flaw has been corrected !


*Thanks !*