Accelerating the digital transformation of the Health sector in the Américas

ALL-IN-ONE
Deployment Guide Without Docker

PAHO Pan American Health Organization World Health Organization Americas Region

ALL-IN-ONE

Deployment Guide Without Docker

Version 1.0

Washington, D.C., 2023

Contents

1. Introduction

The purpose of this guide is to provide instructions on how to deploy the OpenEMR and Telehealth modules on a single server. It is assumed that the user has intermediate knowledge of Linux, Nginx, and PHP-FPM deployment.

1.1 Minimum requirements

The requirements will vary depending on the expected number of users. The most important factor to consider is the expected number of simultaneous teleconsultations. While Jitsi makes every effort to maintain a P2P connection for video calls, it is expected that this will not be possible in all cases. Therefore, some network bandwidth must be allocated for each call. This will vary depending on the user's conditions. In summary, the minimum requirements are as follows:

- A server or VPS with 2 or 4 dedicated cores and at least 2 GB of RAM (as mentioned above, this will vary depending on the usage)
- A domain with 3 subdomains whose A records point to the server's IP address
- SSL certificates for the three subdomains
- Intermediate or advanced Linux knowledge

1.2 Deployment process

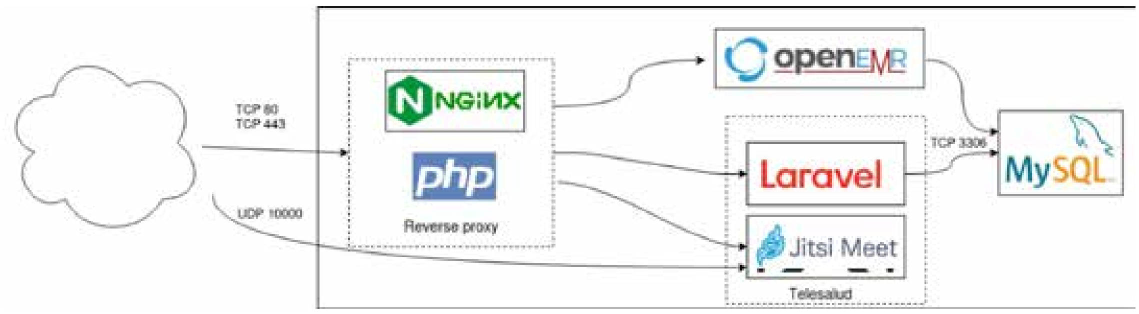
This guide assumes a server with a clean installation of Ubuntu 22.04. A user with sudo privileges is available. This guide only addresses the security of the modules to be deployed. Server security is the responsibility of the administrator and is beyond the scope of this guide. The only ports that need to be open to the internet are the following:

- TCP
  - 80 (http)
  - 443 (https)
- UDP

10000 (Jitsi videobridge, if the telehealth module is installed)

This guide describes the commands necessary for a standard installation of the modules in the environment described above. The commands may vary if the environment is not the same or there are other deployment requirements, so it is desirable to have knowledge of Linux and the technologies used. The OpenEMR and telehealth modules are independent, so it is possible to omit the deployment of one of them if it will not be used.

Because multiple services require the http and https ports, a reverse proxy based on Nginx is used to route traffic based on the requested domain and PHP-FPM to execute PHP scripts if necessary. The following diagram shows the graphical structure of the deployed solution:



First, MySQL (MariaDB can be used instead if desired), Nginx, PHP-FPM, and Composer will be installed because they are dependencies of both modules. Then, the modules will be deployed. Both modules are independent of each other, so one or the other can be omitted if they are not needed.

1.3 Installing MySQL

First, the MySQL server will be installed and a password for root will be configured with the following commands. You must use a strong password for the root user:

```
sudo apt install mysql-server
sudo mysql
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_
password BY 'password for root';
exit;
```

Then, the following command is executed to secure the MySQL installation:

```
sudo mysql_secure_installation
```

The command prompts the user for a series of configuration parameters:

1. A password for root is requested. You must enter a strong password.
2. You are asked if you want to enable password validation, which forces users to use passwords of a certain minimum security level. If you choose yes, the minimum security level to use will be requested next.

3. You are asked if you want to delete anonymous users. You must answer yes (y).
4. You are asked if you want to restrict remote access for the root user. It is recommended to choose yes.
5. You are asked if you want to delete the test database. It is recommended to choose yes.
6. You are asked if you want to update the in-memory access data. Choose yes.

After these steps, MySQL is ready to use. You can test connecting to the server with the command mysql -u root -p, which will prompt for the root password. You should be able to connect successfully.

1.4 Installing NGINX + PHP-FPM + Composer

Next, Nginx will be installed, which will act as a reverse proxy, PHP will be installed to execute the PHP scripts of the modules, and Composer will be installed to manage the dependencies of the modules.

First, Nginx is installed with the following command:

```
sudo apt install nginx
```

This should install the web server. You can test by navigating to the server's IP address and you should see a welcome page.

Then, PHP-FPM will be installed with the necessary extensions for the modules. To do this, the ondrej/php repository is added, which contains all the necessary packages and maintains them over time. The following commands must be executed:

```
sudo apt install software-properties-common ca-certificates lsb-release
apt-transport-https -y
LC_ALL=C.UTF-8 sudo add-apt-repository ppa:ondrej/php
sudo apt update
sudo apt install php8.1-fpm php8.1-mysql php8.1-bcmath php8.1-
xml php8.1-zip php8.1-curl php8.1-mbstring php8.1-gd php8.1-
tidy php8.1-intl php8.1-cli php8.1-soap imagemagick libtiff-tools
php8.1-ldap php8.1-ctype
```

For PHP to function correctly with the modules, some PHP configuration parameters must be modified in the file /etc/php/8.1/fpm/php.ini. The following parameters are detailed to be modified:

- memory_limit: 512M, is the RAM limit to be used by each script

- post_max_size: 30M, is the maximum size of a request body, this is useful for users to upload larger files to the server. You can adjust this parameter to your needs
- max_input_vars: 3000, is the maximum number of variables that a user can enter in a form
- mysqli.allow_local_infile: On, necessary for the correct functioning of the MySQL client

Then, the PHP service must be restarted. It is also desirable to add the user or users who will administer the server to the www-data group to avoid permission problems. This process can be automated with the following commands:

```
sudo sed -i 's/memory_limit = 128M/memory_limit = 512M/' /etc/
php/8.1/fpm/php.ini
sudo sed -i 's/post_max_size = 8M/post_max_size = 30M/' /etc/
php/8.1/fpm/php.ini
sudo sed -i 's/upload_max_filesize = 2M/upload_max_filesize =
30M/' /etc/php/8.1/fpm/php.ini
sudo sed -i 's/;max_input_vars = 1000/max_input_vars = 3000/g' /
etc/php/8.1/fpm/php.ini
sudo sed -i 's/;mysqli.allow_local_infile = On/mysqli.allow_local_in-
file = On/g' /etc/php/8.1/fpm/php.ini
sudo systemctl restart php8.1-fpm
sudo usermod -a -G www-data <usuario>
```

Finally, Composer must be installed with the following commands:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.
php');"
php composer-setup.php
php -r "unlink('composer-setup.php');"
sudo mv composer.phar /usr/local/bin/composer
```

2. Deploying Telehealth Module

The Telehealth module is composed of two submodules. A backend submodule based on Jitsi, which provides the video conferencing service; the second submodule is the frontend based on Laravel, which provides security, control, and additional features to the telehealth module.

2.1 Frontend submodule

First, a database schema, a user, and permissions for the user to use the schema are created. For simplicity, in this guide, the schema and user names are 'telesalud', and the password is 'abc123'. It is recommended to use a strong password in production deployments.

```
mysql -u root -p
CREATE DATABASE telesalud;
CREATE USER 'telesalud'@'localhost' IDENTIFIED BY 'abc123';
GRANT ALL PRIVILEGES ON telesalud.* TO 'telesalud'@'localhost';
FLUSH PRIVILEGES;
exit;
```

NOTE: The word 'localhost' means that the user can only connect from the same computer. If you wish, you can change it to an IP from which you want to connect or '%' if you want to be able to connect from any remote location. It is recommended to allow only local connections and use an SSH tunnel if you want to connect remotely.

The next step is to download the files from the frontend submodule of the repository. In this guide, it is defined that the location of the files will be in a new directory in /opt called 'ops'. The following commands create the 'ops' directory and within it a new directory 'telesalud' that will host the module files, then a git pull operation is performed on the repository of the telesalud module and the directory is entered:

```
cd /opt
sudo mkdir ops
sudo chown -R <usuario>:<usuario> ops
cd ops
git clone https://github.com/ciips-code/ciips-telesalud.git telesalud
cd telesalud
```

You must make a copy of the file .env.example called .env and edit it with a text editor:

```
cp .env.example .env
nano .env
```

Within this file, the lines that need to be customized are commented out. The following lines are necessary for this deployment:

DB_HOST=localhost

DB_DATABASE=telesalud

DB_USERNAME=telesalud

DB_PASSWORD=abc123

Once the .env file has been updated, the Laravel application can be installed and configured:

```
composer install
php artisan key:generate
php artisan migrate
php artisan serve
```

The application will now be available at http://localhost:8000.

- APP_URL: Main URL of the module. In this guide, the domain portaldepacientes.com will be used as the main domain, and this submodule will use the subdomain vc. So the value of the field in this guide will be https://vc.portaldepacientes.com
- LANGUAGE: Language of the module. The accepted values are es and en
- TIMEZONE: Time zone. The possible values can be consulted at https://www.php.net/manual/en/timezones.php
- NOTIFICATION_URL: An URL for the OpenEMR endpoint that will receive notifications about changes in teleconsultations. For this guide, it will be https://openemr.portaldepacientes.com/telehealth/controllers/C_TSalud_Vc.php?action=notify
- NOTIFICATION_TOKEN: In case the endpoint requires security, the token is set
- DB_HOST: In this guide, the database and the module are on the same machine, so you must enter 'localhost'
- DB_DATABASE: Name of the database schema, according to the name entered previously, you must enter 'telesalud'

- DB_USERNAME: Database username that the submodule will use, according to the name entered previously, you must enter 'telesalud'
- DB_PASSWORD: Database password, according to the password entered previously, you must enter 'abc123'
- JITSI_PROVIDER: The frontend submodule has the possibility to work with the backend submodule hosted locally or with the public Jitsi instance, for security reasons the latter option is not recommended. The possible values are jitsi to use the public instance or self to use an instance hosted by you.
- JITSI_BASE_URL: Base URL of the backend submodule, for this guide it will be https://vcbknd.portaldepacientes.com/
- JITSI_APP_ID: Integration identifier with Jitsi, it must match the value configured in Application id of Jitsi. For this guide it will be used 'telesalud'
- JITSI_APP_SECRET: Secret token of the integration with Jitsi, it must match the value configured in Application secret in the Jitsi configuration. For this guide it will be used OafDjrVt8r but it is recommended to use a strong key

Then you must save and exit. Next, the submodule dependencies will be installed with Composer, then a unique key will be created and the database schema will be initialized. You must enter the following commands:

```
composer install
php artisan key:generate
php artisan migrate
```

With the following command, a token is generated to use the module's API. This token must be saved for later use:

```
php artisan token:issue
```

The submodule has directories that must be writable by the user who runs PHP-FPM and Nginx, by default it is www-data. With the following command, the permissions of the necessary directories are changed:

```
sudo chown -R www-data:www-data bootstrap/cache storage/
logs storage/framework storage/app
```

Finally, you must configure Nginx to use the submodule. To do this, create a new file in /etc/nginx/sites-enabled called 'telesalud.conf'. The content of this gist on GitHub must be inserted, modifying the parameters as needed, which are commented in the gist. The following data must be modified obligatoriamente:

```
server_name: The domain or IP address of the server where the
module will be hosted.
root: The path to the directory where the module files are located.
index: The name of the index file that will be served when accessing
the module.
ssl_certificate: The path to the SSL certificate file.
ssl_certificate_key: The path to the SSL certificate key file.
Once the configuration file has been created, Nginx must be restarted.
```

To access the module, you can navigate to the domain or IP address of the server in your browser.

- server_name: The domain name that points to the submodule. For this guide, you should enter 'vc.portaldepacientes.com'. **NOTE:** This parameter is located twice in the file, so you must modify both.
- root: The path that points to the public folder of the submodule. For this guide, you should enter '/opt/ops/telesalud/public'.
- ssl_certificate: The location of the certificate file, which you have previously copied to the server. For this guide, you should use '/etc/ssl/private/vc.crt'.
- ssl_certificate_key: The location of the certificate key file, which you have previously copied to the server. For this guide, you should use '/etc/ssl/private/vc.key'.

To edit the file, enter the following command:

```
sudo nano /etc/nginx/sites-enabled/telesalud.conf
```

Finally, you must check that the Nginx configuration is correct with the command sudo nginx -t. If the output of the command indicates that it is correct, you must restart the Nginx service with the following command:

```
sudo /etc/init.d/nginx reload
```

2.2 Backend submodule

Before deploying the submodule, you need to add a dependency and enable the universe repository in the system:

```
sudo apt install gnupg2
sudo apt-add-repository universe
sudo apt update
```

Because Jitsi has its own repository, you need to add the keys to use it, then add the repository and also install lua5.2, which is a necessary dependency. You need to enter the following commands:

```
sudo curl -sL https://prosody.im/files/prosody-debian-packages.key
-o /etc/apt/keyrings/prosody-debian-packages.key
echo "deb [signed-by=/etc/apt/keyrings/prosody-debian-packages.
key] http://packages.prosody.im/debian $(lsb_release -sc) main" |
sudo tee /etc/apt/sources.list.d/prosody-debian-packages.list
sudo apt install lua5.2
curl https://download.jitsi.org/jitsi-key.gpg.key | sudo sh -c 'gpg
--dearmor > /usr/share/keyrings/jitsi-keyring.gpg'
echo 'deb [signed-by=/usr/share/keyrings/jitsi-keyring.gpg] https://
download.jitsi.org stable/' | sudo tee /etc/apt/sources.list.d/jitsi-sta-
ble.list > /dev/null
sudo apt update
```

The following command downloads and installs Jitsi:

```
sudo apt install jitsi-meet
```

This will start an installation wizard that will request a series of configuration data:

1. Domain to use: Enter the subdomain assigned for the backend, for this guide it will be vcbknd.portaldepacientes.com
2. Certificate to use: Enter one of the 3 options
   a. *Let's encrypt certificates: Use certificates obtained automatically from Let's encrypt. This option is recommended only if you do not have a certificate for the subdomain.*
      i. "If you select this option, you will be prompted for an email address to register with Let's encrypt."
   b. *I want to use my own certificate: Use previously obtained certificates. Recommended option, but you must previously obtain paid certificates and copy them to the server.*
      i. "If you select this option, you will be prompted for the path to the certificate file and key."
   c. *Generate a new self-signed certificate: Generate a self-signed certificate. This option is only viable for development environments.*

The Jitsi installer modifies the Nginx configuration, so it is necessary to check that it has been correct with the command sudo nginx -t. If the output of the command indicates that it is correct, the Nginx service must be restarted with the following command:

```
sudo /etc/init.d/nginx reload
```

Then you must install jitsi-meet-tokens. This Jitsi plugin restricts access to the platform to users who have a valid JWT token to prevent misuse of the service. You must enter the following command:

```
sudo apt install jitsi-meet-tokens
```

This will start an assistant that will request data to configure:

1. *Application id: Must match the value configured in the JITSI_APP_ID parameter of the frontend .env file*
2. *Application secret: Must match the value configured in the JITSI_APP_SECRET parameter of the frontend .env file*

Optionally, you can change the watermark logo that appears on the Jitsi screen. The following command replaces the original watermark with the OPS logo:

```
sudo cp /opt/ops/telesalud/jitsi-docker/resources/watermark.png /
usr/share/jitsi-meet/images/watermark.svg
```

After these steps, the telehealth module is already up and running. You can verify it using Postman with the examples.

3. Deploying OpenEMR

First, you need to create a database schema, a user, and grant the user permissions to use the schema. For simplicity, in this guide we will use 'openemr' as the schema and user name, and 'abc123' as the password. It is recommended to use a strong password in production deployments.

```
mysql -u root -p
CREATE DATABASE openemr;
CREATE USER 'openemr'@'localhost' IDENTIFIED BY 'abc123';
GRANT ALL PRIVILEGES ON openemr.* TO 'openemr'@'localhost';
FLUSH PRIVILEGES;
exit;
```

NOTE: The word "localhost" means that the user can only connect from the same computer. If you want, you can change it to an IP address from which you want to connect or '%' if you want to be able to connect from any remote location. It is recommended to only allow local connections and use an SSH tunnel if you want to connect remotely.

The next step is to download the OpenEMR module files from the repository. As with the telehealth module, it is defined that the location of the files will be in a new directory in /opt called 'ops'. The following commands create the 'ops' directory (omit this step if the telehealth module has already been deployed) and within it a new 'openemr' directory that will host the module files, then a git pull operation is performed on the OpenEMR module repository and the directory is entered:

```
cd /opt
sudo mkdir ops
sudo chown -R <usuario>:<usuario> ops
git clone https://github.com/ciips-code/openemr-telesalud.git openemr
cd openemr
```

Next, you need to configure the database connection parameters. The following commands create a copy of the generic connection file and edit the new file:

```
cd openemr/sites/default
cp sqlconf.sample.php sqlconf.php
nano sqlconf.php
```

The following parameters must be modified:

- $host: The address of the database server. For this guide, this will be localhost.
- $port: The port on which the database server listens. For this guide, this will be 3306.
- $login: The username of the database user for the module. For this guide, this will be openemr.
- $pass: The password of the database user for the module. For this guide, this will be abc123.
- $dbase: The name of the database schema for the module. For this guide, this will be openemr.
- $config: This must be set to 1 to skip the OpenEMR installation wizard.

After saving the file, you must create a copy of the .env.example file called .env in the root of the module.

```
cd ../..
cp .env.example .env
nano .env
```

In this file, you must modify the database connection data with the same data as the previous file. In addition, you must configure other parameters:

- VC_API_URL: The base URL of the frontend sub-module of the telehealth module. For this guide, this will be https://vc.portaldepacientes.com.
- VC_API_PORT: The port of the frontend sub-module of the telehealth module. For this guide, this will be 443.
- VC_API_TOKEN: A token previously obtained to use the API of the telehealth module.

With the following command, the module dependencies will be installed with Composer:

```
composer install
```

OpenEMR requires NPM to recreate some assets. You must install NVM and then NPM in its version 16. Then, the NPM dependencies for the project are installed and the assets are recreated:

```
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh
| bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
nvm install 16
nvm use 16
npm install
npm run build
```

Next, you need to initialize the database schema for the module with the pre-existing tables and data. These are located in the ciips/docker/sql directory.

```
cd ciips/docker/sql
```

You must execute each of the sql and sql.gz files in order according to the date present in the file name.

To execute sql.gz files, modify and enter the following command:

```
zcat archivo.sql.gz | mysql -u openemr -p openemr
To execute sql files, modify and enter the following command:
mysql -u openemr -p openemr < archivo.sql
```

NOTE: you must use the username and schema previously created. For this guide, both are 'openemr'.

The module has directories that must be writable by the user that executes PHP-FPM and Nginx, by default is www-data. With the following command, the permissions of the necessary directories are changed:

```
cd ../../../
sudo chown -R www-data:www-data sites/default/documents
```

Finally, you need to configure Nginx to use the module. To do this, create a new file in /etc/nginx/sites-enabled called openemr.conf. You must insert the content of this gist on GitHub, modifying the parameters that are necessary, which are commented in the gist. You must modify the following data obligatorily:

- server_name: domain that points to the submodule, for this guide, enter 'openemr.portaldepacientes.com'. NOTE: this parameter is found twice in the file, both must be modified

- root: path that points to the public folder of the submodule. For this guide, enter '/opt/ops/openemr/public'
- ssl_certificate: location of the certificate file, previously copied to the server. For this guide, it uses '/etc/ssl/private/openemr.crt'
- ssl_certificate_key: location of the certificate key file, previously copied to the server. For this guide, it uses '/etc/ssl/private/openemr.key'

To edit the file, enter the following command:

```
sudo nano /etc/nginx/sites-enabled/openemr.conf
```

Finally, you need to check that the Nginx configuration is correct with the command sudo nginx -t. If the output of the command indicates that it is correct, you need to restart the Nginx service with the following command:

```
sudo /etc/init.d/nginx reload
```

With these steps, the OpenEMR module is already deployed. You can check it by navigating to the domain configured for this case.

4. Useful links

- https://nginx.org/en/docs/
- https://dev.mysql.com/doc/
- https://github.com/ciips-code/
- https://github.com/ciips-code/ciips-telesalud/
- https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-quickstart/
- https://laravel.com/docs/9.x
- https://www.open-emr.org/wiki/index.php/OpenEMR_Wiki_Home_Page
- https://github.com/nvm-sh/nvm