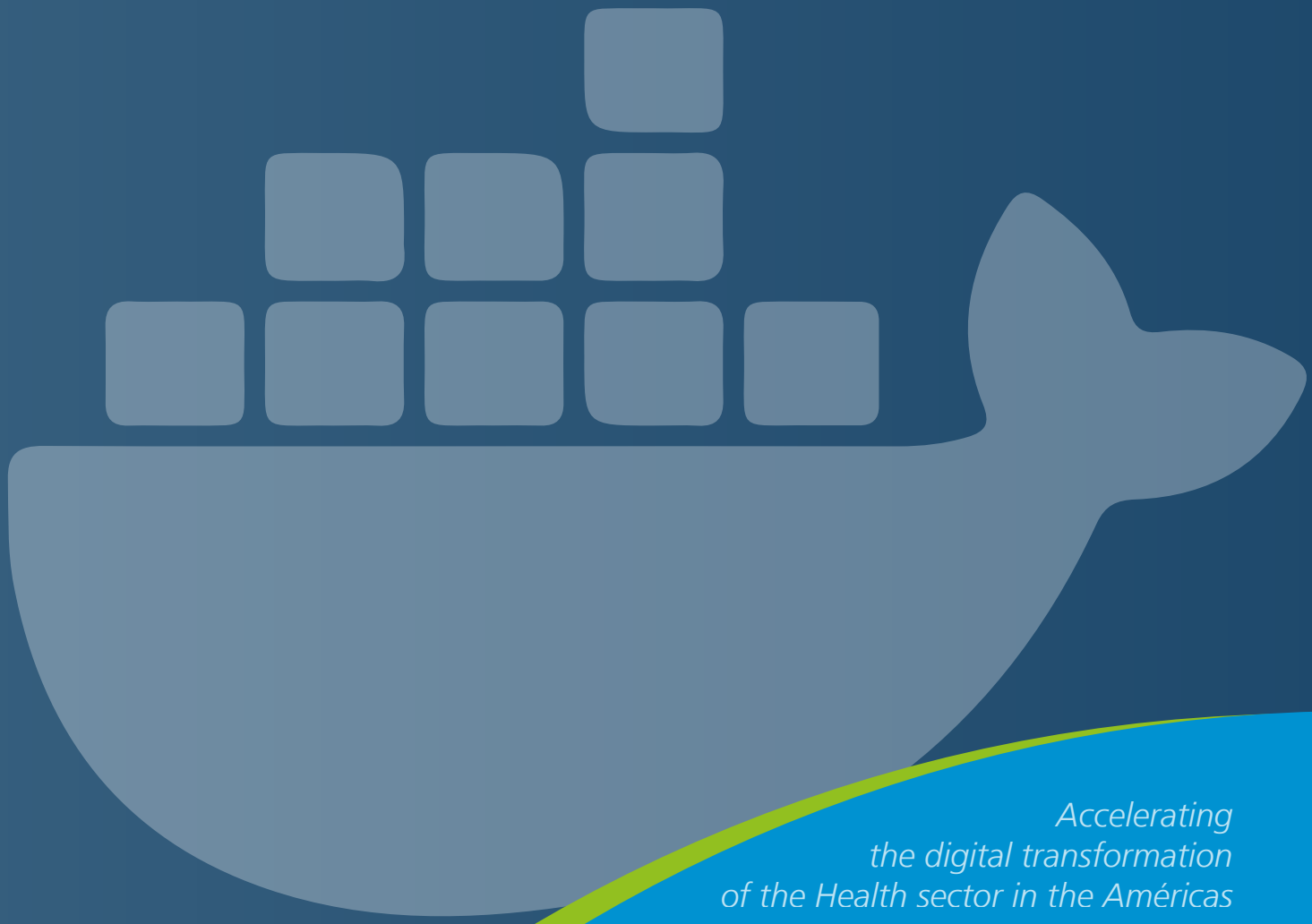


Accelerating
the digital transformation
of the Health sector in the Américas

ALL-IN-ONE
Deployment Guide Using Docker

PAHO
Pan American Health Organization
World Health Organization
Americas Region

ALL-IN-ONE

Deployment Guide Using Docker

Version 1.0

Washington, D.C., 2023

PAHO Pan American Health Organization World Health Organization Americas Region

Contents

1. Introduction

All In One Platform is a Telehealth and Clinical Record solution based on free and open-source software (FOOS) to be shared by PAHO as a Digital Public Good.

This solution is a combination of adaptation of the electronic clinical record system OpenEMR (www.open-emr.org) and the creation of a video-encounter module based on the open-source videoconferencing technology of Jitsi (www.jitsi.org) with interoperability interfaces between them for the communication and saving of the video-encounter clinical information.

1.1 Overview

The solution was designed as a platform with two independent modules that can be implemented separately and integrated into the country's existing system.

One module is an electronic clinical record with schedules management, patient index and structured clinical documents like SOAP and vital signs register. And the other module is the telehealth component that allows remote connection between health providers and patients with a clinical notes register and the capacity to exchange attached documents.

The aim of this document is to guide the deployment process of the Clinical Record and Telehealth modules on a single server. It is assumed as a precondition that the technical team have basic knowledge of docker and docker-compose.

More information: https://docs.docker.com/.

2. Getting Started

2.1 Set-up Considerations

- A server or VPS with 2 or 4 cores, 2 GB of RAM.
    - These requirements will scale depending on the number of concurrent users.
- A domain with 3 subdomains pointing to the IP of the server.

3. Deployment Process

3.1 First steps

In this guide, an Ubuntu-based server is taken with docker, docker-compose and git already installed as the scenario.

At the same time, the relevant security measures were taken.

The user that will be used will be ***ubuntu***.

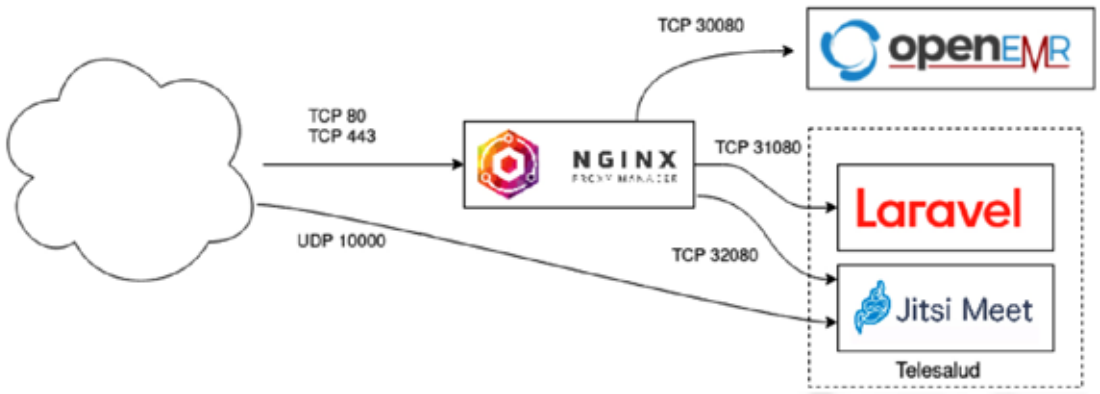
The open ports on the server are:

- TCP
  - 22 (ssh, if it necessary)
  - 80 (http)
  - 443 (https)
- UDP
  - 10000 (Jitsi video bridge, in case of installing a telehealth module)

A reverse proxy based on Nginx Proxy Manager is used that routes the traffic based on the requested domain since multiple modules require HTTP and HTTPS ports

The following diagram shows the structure of the deployed solution.

Figure 1. All in one diagram



The first step is to create the directory that will host all the files.

Unless only one user is maintaining the solution it is recommended that it not be a home directory.

/opt/ops directory will be used in this guide.

We create them and assign the necessary permissions.

In this guide, we use ***ubuntu*** group. However, you can create another group for better permission control.

```
cd /opt
mkdir ops
sudo chown -R ubuntu: ubuntu ops
cd ops
```

3.2 Telehealth Module Deployment

Telehealth Module has two sub-modules:

- **Backend Sub-module:** based on Jitsi and gives the videoconference service.
- **Laravel-based Frontend Sub-module:** provides security, control and extra features to the telehealth module.

3.2.1 Frontend sub-module

The first step is to clone the module repository into a new directory name telehealth.

```
git clone https://github.com/ciips-code/ciips-telesalud.git telehealth
cd /opt/ops
telehealth cd
```

Then, a copy of the ***.env.example file*** is made called ***.env***

cp .env.example .env

With a text editor, the ***.env file is edited***

nano .env

Inside this file are line comments that must be customized as detailed below.

- *WEB_LISTEN_PORT:* The port that will listen to HTTP requests, according to the diagram, the port will be 31080
- *WEB_HTTPS_LISTEN_PORT:* Port that will listen to HTTPS, in this deployment it will not be used; however by convention 31443 must be used.

- DB_LISTEN_PORT: Port to connect to the MySQL database of the module, for security it must be protected from the outside and then it can be omitted. By convention 31006 should be used.
- *portaldepacientes.com* will be used as the main domain, and this sub-module will use the ***vc subdomain***. Therefore, the value of the field in this guide will be https://vc.portaldepacientes.com.
- LANGUAGE: Language of the module, the accepted values is es and *en*.
- TIMEZONE: Time zone, the possible values can be consulted at https://www.php.net/manual/en/timezones.php
- NOTIFICATION_URL: A URL for the OpenEMR endpoint that will receive notifications about changes in the tele-encounter. For this guide it will be https://demo.portaldepacientes.com/endpoint
- NOTIFICATION_TOKEN: In case the endpoint requires security, the token is established.
- DB_DATABASE: Name of the database schema, modifying it is optional.
- DB_USERNAME: Database user name that the sub-module will use, modifying it is optional.
- DB_PASSWORD: Database password, use a strong key.
- JITSI_PROVIDER: The frontend sub-module has the possibility to work with the locally hosted backend sub-module or with the public ***Jitsi*** instance, for security reasons this last option is not recommended. Possible values are jitsi to use the public instance or ***self*** to use an instance you host.
- JITSI_BASE_URL: base URL of the backend sub-module, for this guide https://vc-bknd.portaldepacientes.com/ will be used
- JITSI_APP_ID: Jitsi integration identifier must match the value configured in JWT_APP_ID in the Jitsi configuration. ***Telehealth*** will be used for this guidance
- JITSI_APP_SECRET: Jitsi integration secret token, must match the value configured in JWT_APP_SECRET in the Jitsi configuration. ***OafDjrVt8r*** will be used for this guide. It is recommended to use a strong password.

Edit the ***docker-compose.yml file***. Change the MySQL root password on line 30.

HTTPS will not be used internally in this deployment, however, a certificate and key must be created to avoid errors in the sub-module. To do this, the following command must be executed. It will ask you some questions, which can be omitted by pressing enter.

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout docker-config/cert.key -out docker-config/cert.crt
```

The sub-module must be run with the command docker-compose up -d. The first time it is run this process will take a few minutes.

Once it has been executed, a series of commands must be executed. These commands should only be executed the first time the sub-module is raised:

docker-compose exec app composer install. To install sub-module dependencies
docker-compose exec app php artisan key:generate. To generate a unique key.
docker-compose exec app php artisan migrate. To create the data-base structure.
docker-compose exec app php artisan token:issue. To the creation of an API access token, the resulting value must be configured in the OpenEMR *.env file*. You need to save it for later.

3.2.2 Backend sub-module

This sub-module is located in the

jitsi-docker directory
cd jitsi-docker
.env.example file should be made called *.env*
cp .env.example .env

Edit this file with the following customizations:

HTTP_PORT: Port that listens for http requests. Following the guide it will be 32080
HTTPS_PORT: Port that listens for https requests. It is not used in this deployment but by convention 32443 is configured
TZ: Time zone, it is configured in the same way as the frontend sub-module
PUBLIC_URL: URL of the backend sub-module. For this guide it will be https://vcbknd.portaldepacientes.com
JWT_APP_ID: Frontend integration identifier, must match the previously configured
JWT_APP_SECRET: Frontend integration secret key, must match the previously configured

Finally, you must run docker-compose up -d. Like the frontend, the first boot takes a few minutes.

3.3 OpenEMR Deployment

The first step is to clone the openemr-telesalud repository in your development environment.

These commands clone the repository to a new directory named openemr and position itself inside it.

```
cd /opt/ops
git clone https://github.com/ciips-code/openemr-telesalud.git openemr
cd openmr
```

In the *ciips/docker directory* there are the necessary files for the deployment of the OpenEMR module, so it must be located in that directory

```
cdciips/docker
```

Copy the file ***.env.example*** to ***.env and docker-compose.prod.yml to docker-compose.yml***

```
cp .env.example .env
cp docker-compose.prod.yml docker-compose.yml
```

.env file customizing the following parameters:

COMPOSE_PROJECT_NAME: Base name for the project within docker

MYSQL_ROOT_PASSWORD: Password for the root user in the MySQL container, use an extra strong password

MYSQL_PORT: Port where the MySQL container will listen, for security reasons this port must be protected from the outside. For this guide, port 30006 will be used.

MYSQL_DATABASE: Name of the database schema where the module data will be stored

MYSQL_USER: MySQL username that the module will use

MYSQL_PASSWORD: Password for the MySQL user of the module. Choose a strong password

TELEHEALTH_BASE_URL: base URL of the frontend sub-module of the telehealth module. For this guide, https://vc.portaldepacientes.com will be used

TELEHEALTH_PORT: Telehealth module listening port. For this guide 443 is used

TELEHEALTH_API_TOKEN: Token previously obtained after running docker-compose exec app php artisan token:issue in frontend sub-module

OPENEMR_PORT: Port where OpenEMR will listen for http requests. For this guide, port 30080 will be used.

Run the module with the command ***docker-compose up -d***

3.4 Reverse proxy

Nginx Proxy Manager (from now NPM) will be used as a reverse proxy solution in this document.

In the case of using another solution, you need to consider that the backend of the telehealth module requires Websockets support, so this type of traffic must also be redirected.

NPM will be deployed to a directory called *proxy* in the home directory.

```
cd /opt/ops
mkdir proxy
proxy cd
```

Within this directory, a ***docker-compose.yml*** file is created with the contents of the following link https://pastebin.com/LbhRq22V (due to the nature of YAML, it is impractical to copy the contents from this document). If you want to change the NPM web administration port, you can customize line 10, the default port is 81. If you want to customize, you must change the first number of the line. For example, if you want to use port 3081, edit the line like this: - ***'3081:81'*** .

You should run the command ***docker-compose up -d*** and wait.

3.5 Shared network between containers configuration

For the communication between NPM and the modules, must be created a docker network and relevant containers connected to it. The first thing to do is create a new network, which will be called *frontend*. It is created with this command:

```
docker network create frontend
```

Subsequently, the containers that respond to HTTP requests must be connected and are the following:

- Proxy: nginx container
- openemr: openemr container
- Frontend teleconsultations: web container
- Backend teleconsultations: web container

The name of each container must be obtained with the docker ps command.

The output of this command is a list of all active containers. In this case, the most important point is the NAMES column, where the names of the containers are listed.

Each name is integrated with the directory name of the ***docker-compose.yml*** that originated it, followed by a hyphen, then the name of the service (openemr, nginx, web), followed by another hyphen and an instance number.

Within this directory, a ***docker-compose.yml*** file is created with the contents of the following link http://pastebin.com/LbhRq22V (due to the nature of YAML, it is impractical to copy the contents of this document). If you want to change the NPM web administration port, you can customize line 10, the default port is 81. If you want to customize, you must change the first number of the line. For example, if you want to use port 3081, edit the line like this: -`3081:81´.

You should run the command *docker-compose up -d* and wait

For this guide, the containers are: (in red those that will be added to the *frontend* network)

- proxy-proxy-1
- jitsi-docker-jvb-1
- jitsi-docker-jicofo-1
- jitsi-docker-prosody-1
- jitsi-docker-web-1
- telehealth-web-1
- telehealth-app-1
- telehealth-database-1
- docker-openemr-1
- docker-mysql-1

To add each container to the network, use the docker network add frontend ***<container_name> command*** .

In this case, the commands are the following:

```
docker network add frontend proxy-proxy-1
docker network add frontend jitsi-docker-web-1
docker network add frontend telehealth-web-1
docker network add frontend docker-openemr-1
```

3.6 NPM Configuration

To configure NPM, you must open a browser and go to the IP or domain of the server and the chosen port.

For this guide, it is 81. Therefore the resultant URL is http://portaldepacientes.com:81/.

The first phase is fundamental to avoiding certificate problems, so it used HTTP, not HTTPS.

At the login, the default username and password must be entered:

- **User:** admin@example.com
- **Password:** change me

You have to choose a strong password when the system asks you to.

After changing the data, you will enter the main NPM screen. Go to the Hosts menu, Proxy Hosts option to configure access to the modules.

On the next screen, the modules will be listed. Initially, this list will be empty. A new Host must be added by pressing the Add Proxy Host button. A popup will be open where you must configure the parameters for each service:

- Details tab
  - *Domain names*: the domain for the service is configured in this field, for example vc.portaldepacientes.com
  - *Forward* ***Hostname/IP***: the container´s name must be specified in the same way that was added to the network in previous steps. The only exception is the ***proxy-proxy-1 container.***
  - *Forward Port*: port to forward, always enter 80
  - *Block common exploits*: always enable
  - *Websockets support*: enable for ***jitsi-docker-web-1***
- *SSL* tab
  - *SSL Certificate*: Select the "Request a new SSL certificate" option. Automatically, a Let's Encrypt certificate will be requested.
  - *Force SSL*: Enable so that https is always used
  - *HTTP/2 Support* : Enable to enable better features

Then, save and wait a few seconds. Navigate to the new domain to verify that the configuration was correct.

Details of the three services that must be created and configured according to the values in this guide:

- OpenEMR
  - *Details* tab
    - *Domain names :**portaldepacientes.com www.portaldepacientes.com***
    - *Forward Hostname/IP:* ***docker-openemr-1***
    - *Forward Port:* 80
    - *Block common exploits:* Enable
  - *SSL* tab
    - *SSL Certificate:* ***Request a new SSL certificate***
    - *Force SSL:* Enable
    - *HTTP/2 Support:* Enable

- frontend telehealth
  - *Details* tab
    - *Domain names:* ***vc.portaldepacientes.com***
    - *Forward Hostname/IP:* ***telesalud-web-1***
    - *Forward Port:* 80
    - *Block common exploits:* Enable
  - *SSL* tab
    - *SSL Certificate :* ***Request a new SSL certificate***
    - *Force SSL:* Enable
    - *HTTP/2 Support:* Enable

- telehealth backend
  - *Details* tab
    - *Domain names :* ***vcbknd.portaldepacientes.com***
    - *Forward Hostname/IP:* ***jitsi-docker-web-1***
    - *Forward Port:* 80
    - *Block common exploits:* Enable
    - *Websockets Support:* Enable
  - *SSL* tab
    - *SSL Certificate:* ***Request a new SSL certificate***
    - *Force SSL:* Enable
    - *HTTP/2 Support:* Enable

Following all these actions, the module's deployment is finished and ready to work.

For the installation control, you must navigate to the URL defined for the OpenEMR module. For this guide is https://portaldepacientes.com .

You will see the login screen. The default login credentials are:

- **Username:** admin
- **Password:** AdminOps2023**

These credentials are standard for all docker installations. It is recommended to change them for protection.

4. Reference documentation

Official Docker Resources
https://docs.docker.com/

Official Laravel Documentation
https://laravel.com/docs/9.x

Official Jitsi Documentation
https://laravel.com/docs/9.xhttps://jitsi.github.io/handbook/docs/intro/

Repository of the Electronic Clinical Record Module
https://github.com/ciips-code/openemr-telesalud

Platform Repository
https://github.com/ciips-code/
https://github.com/ciips-code/ciips-telesalud

Official OpenEMR Documentation
https://www.open-emr.org/wiki/index.php/OpenEMR_Wiki_Home_Page

PAHO
Pan American Health Organization
World Health Organization
Americas Region

www.paho.org