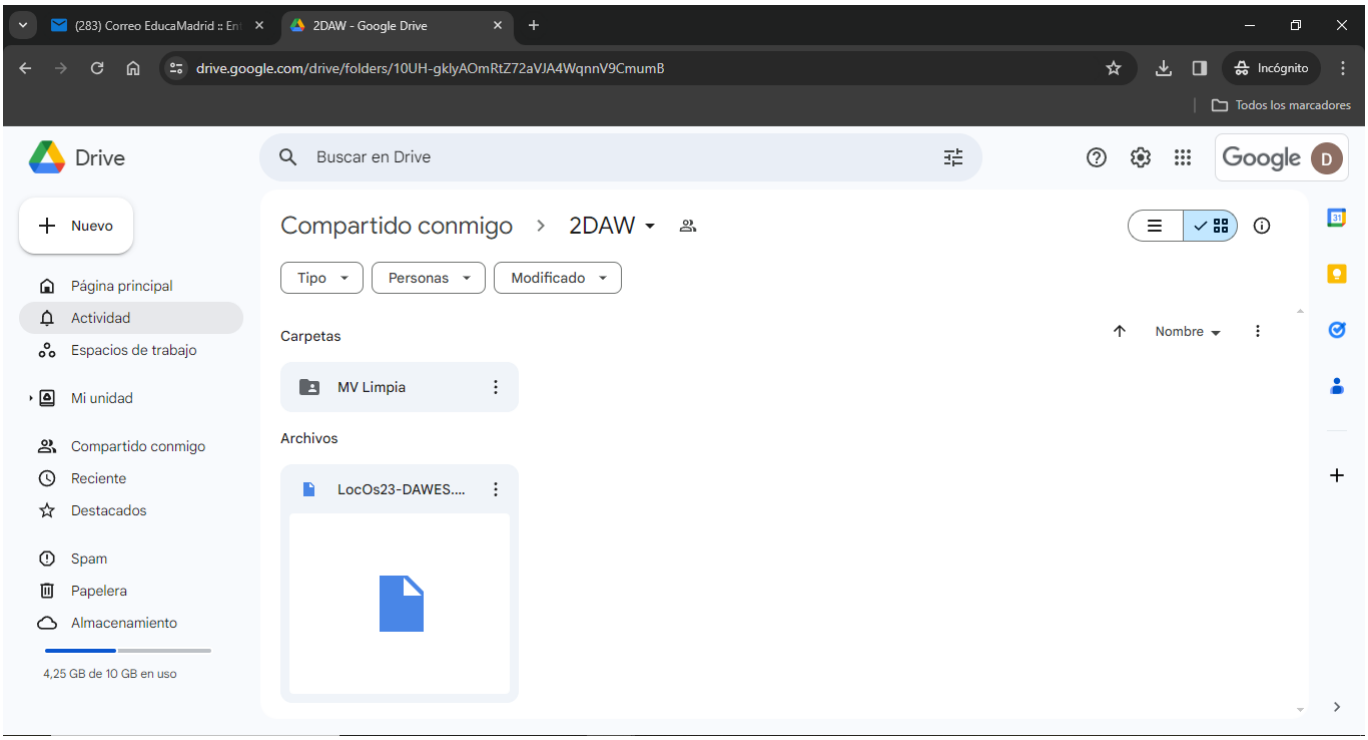
Indice de contenidos:

1. Máquina virtual

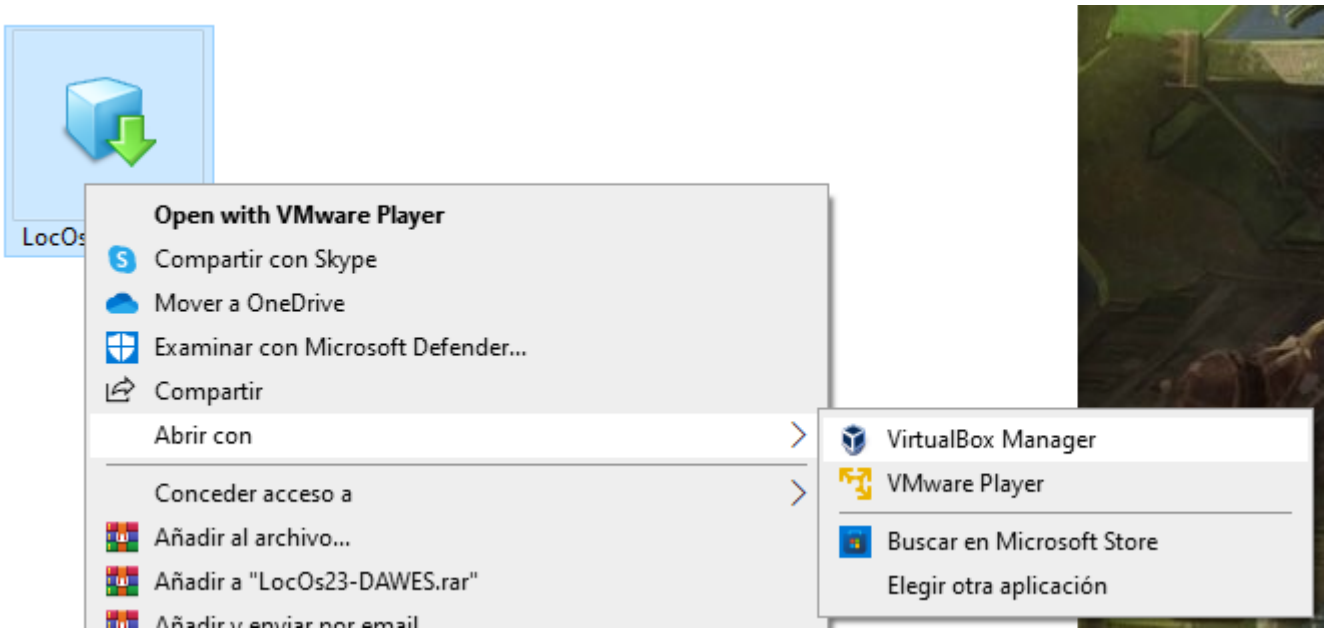
1.1 Adquirir la ova de la máquina virtual

Para conseguir la OVA nos vamos al repositorio que corresponda, como ahora no estoy en el examen toca meterse en Google Drive con la cuenta de educa.madrid.org, una vez en drive seleccionamos a la izquierda "Compartido conmigo" -> "2DAW" y luego descargamos la LocOs23 DAWES OVA, la de 4,2 GB, viene con docker y php instalados, el usuario se llama "usuario" y la contraseña de administrador es "usuario"

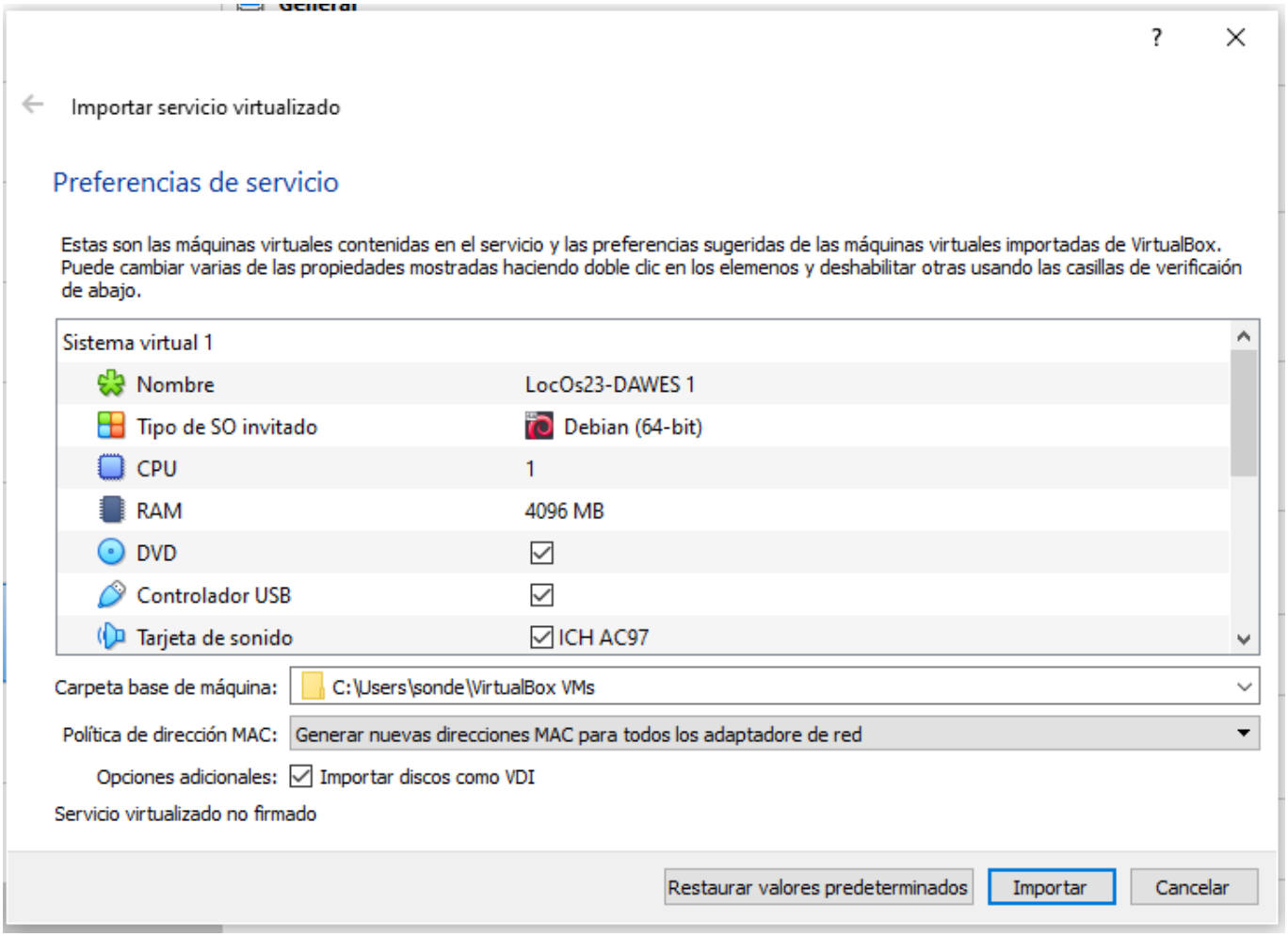


1.2 Instalar la máquina virtual

Cuando tengais la OVA teneis que hacer click derecho -> Abrir con -> virtualbox

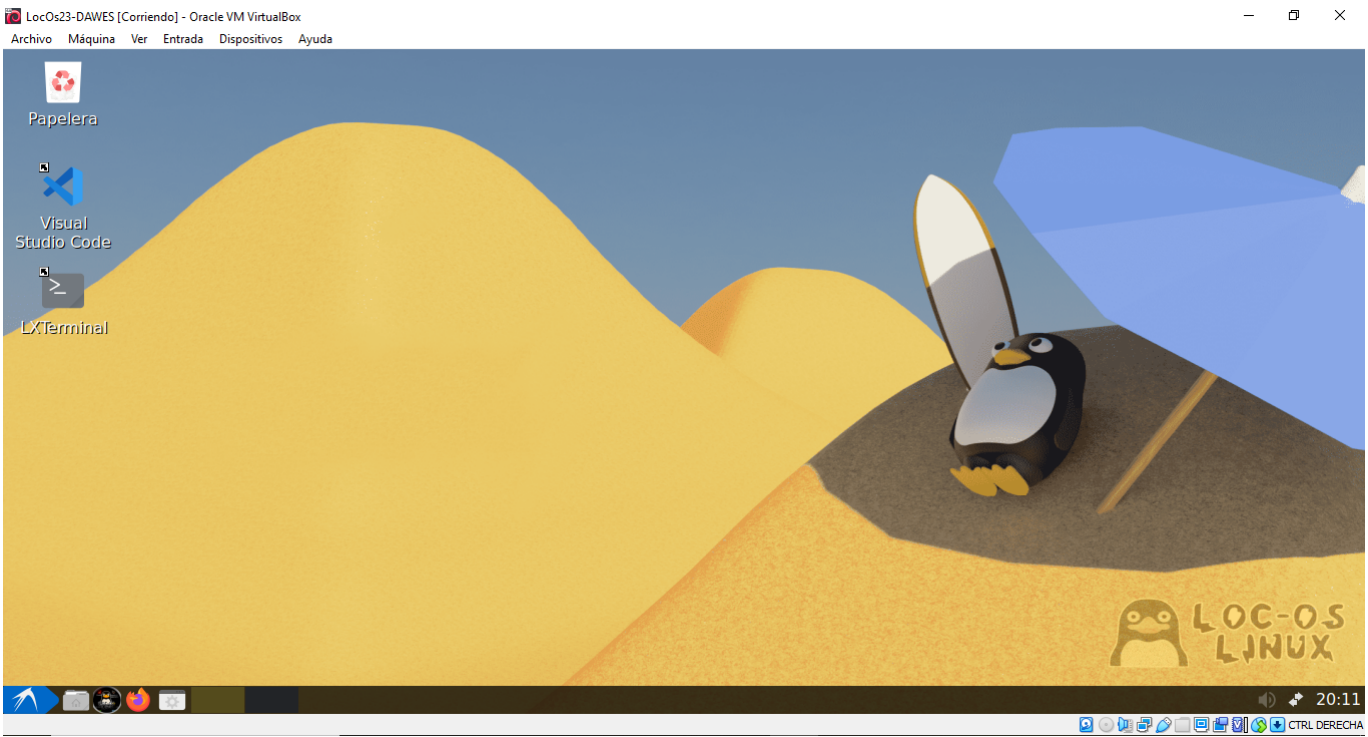


Os aparecera el menú de VirtualBox, dejais todo como está excepto que en "Politica de direcciones MAC" abrís el desplegable y elegís generar nuevas MAC para todos los adaptadores. Le dais a importar.



1.3 Abrir la máquina virtual

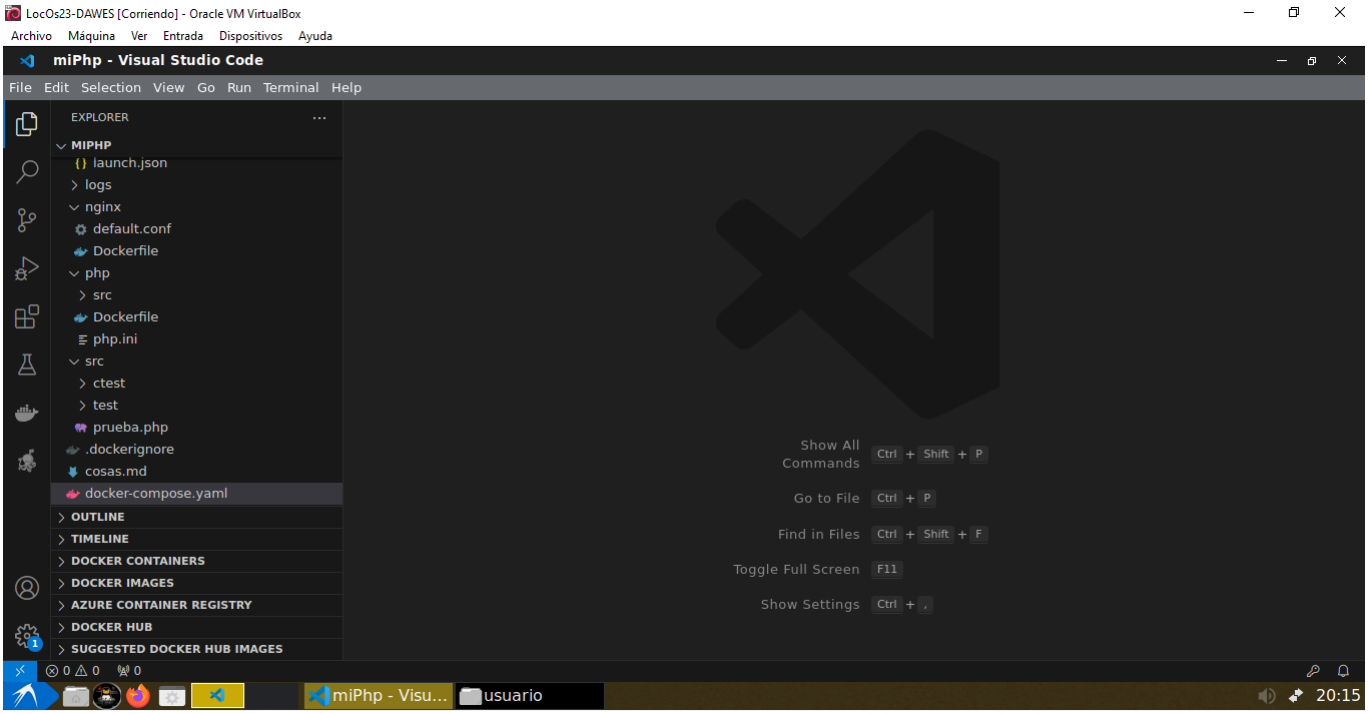
La abrimos con normalidad y según nos metemos ha de verse algo así:

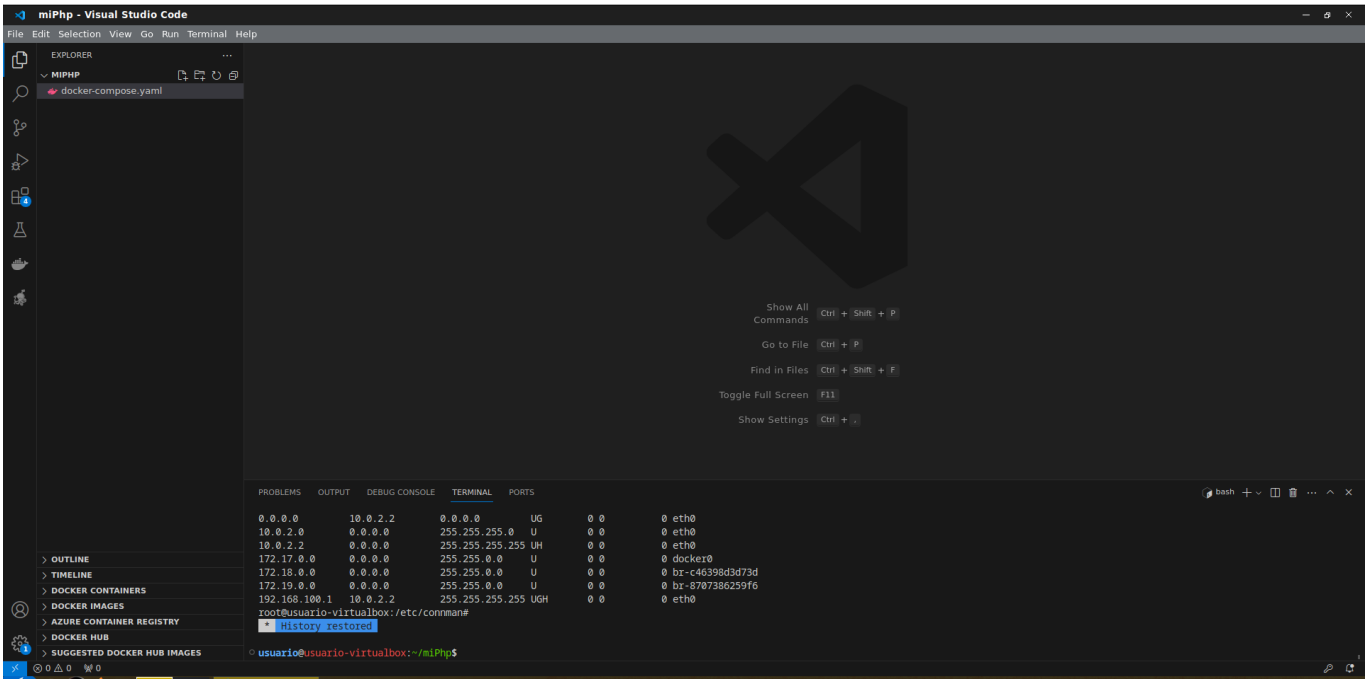


2.Docker

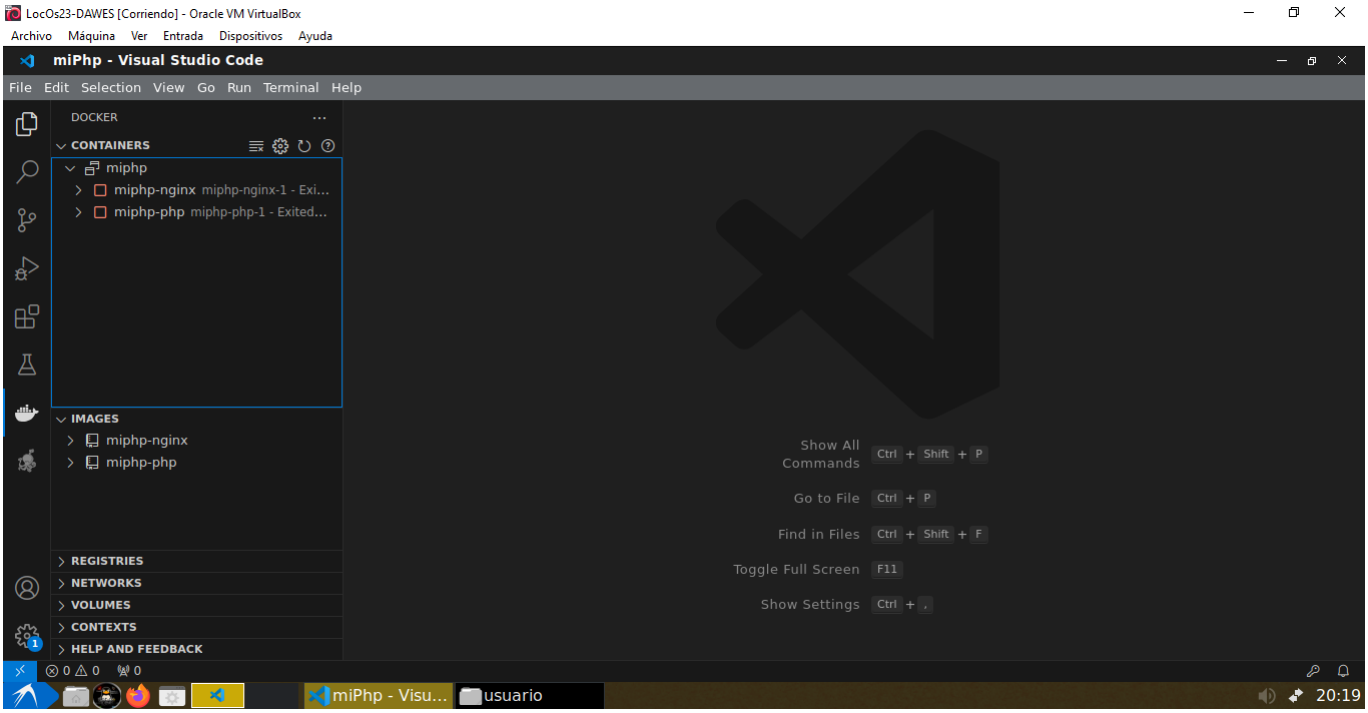
2.1 Preparar docker

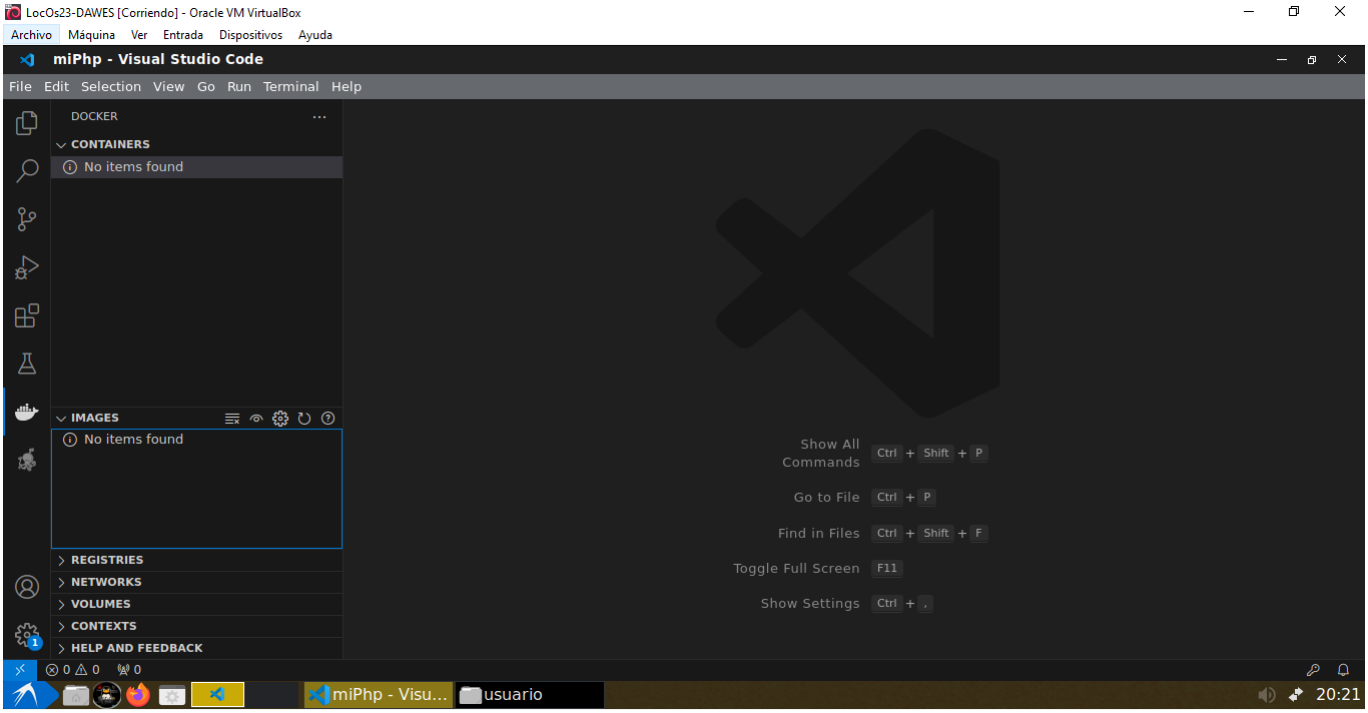
Abrimos el visual y nos deberiamos encontrar en una carpeta llamada "miPHP", borramos todo lo que halla en ella menos el "docker-compose.yaml"





Luego entramos en el apartado de Docker y borramos todo, los contenedores y las imágenes:





Ahora volvemos y en la carpeta miPHP el "docker compose.yaml" que teniamos vamos a sustituir su contenido por el siguiente:

```
version: '2'

services:
  mariadb:
    image: docker.io/bitnami/mariadb:11.2
    ports:
      - '3306:3306'
    expose:
      - '3306'
    env_file:
```

```
      - ".env"
    environment:
      # ALLOW_EMPTY_PASSWORD is recommended only for development.
      - ALLOW_EMPTY_PASSWORD=yes
      - MARIADB_USER=${MARIADB_USER}
      - MARIADB_DATABASE=${MARIADB_DATABASE}
      - MARIADB_PASSWORD=${MARIADB_PASSWORD}
  myapp:
    tty: true
    image: docker.io/bitnami/laravel:10
    labels:
      kompose.service.type: nodeport
    ports:
      - '8000:8000'
    env_file:
      - ".env"
    environment:
      - DB_HOST=mariadb
      - DB_PORT=3306
      - DB_USERNAME=${MARIADB_USER}
      - DB_DATABASE=${MARIADB_DATABASE}
      - DB_PASSWORD=${MARIADB_PASSWORD}
    volumes:
      - './Proyecto-Julian:/app'
    depends_on:
      - mariadb
```

2.2 Hacer el .env del docker

Crearemos en miPHP un archivo llamado ".env" con el siguiente contenido:

```
# MaribaDB - Desarrollo
MARIADB_VERSION=8.0.21
MARIADB_HOST=mariadb
MARIADB_DATABASE=bitnami_myapp
MARIADB_USER=bn_myapp
MARIADB_PASSWORD=secret
```

---

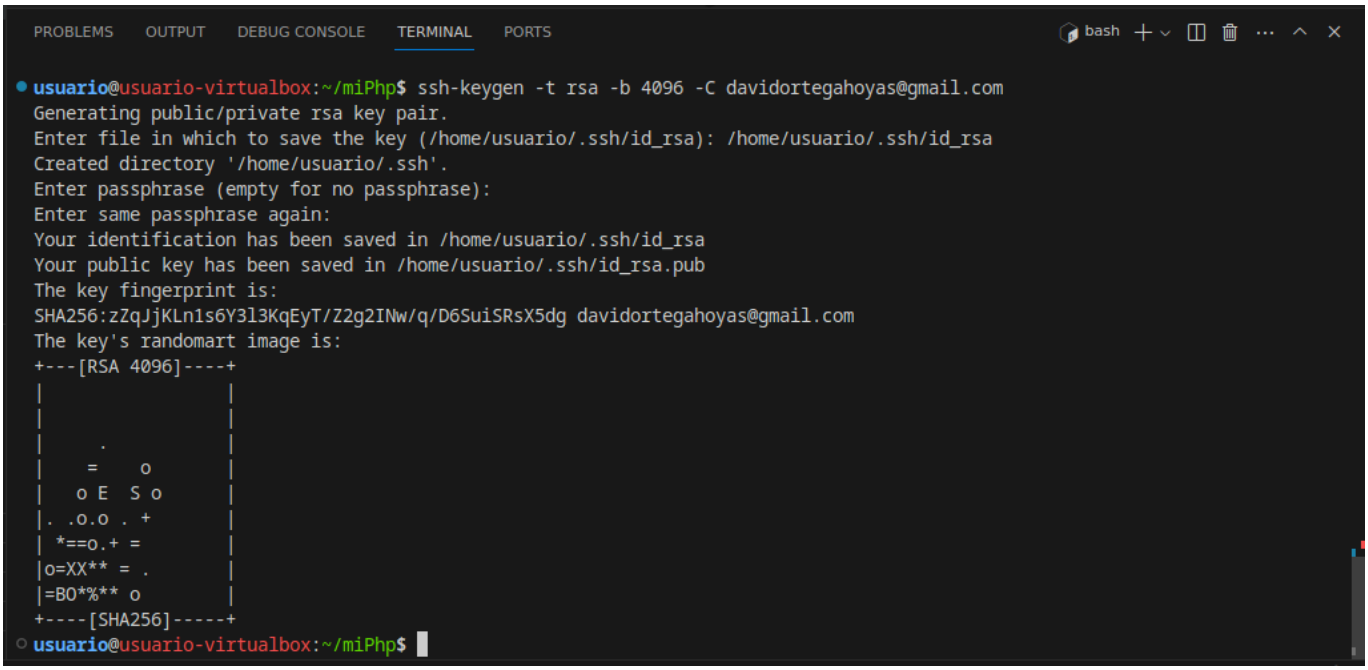
3. Clonar el proyecto

3.1 SSH

Para poder clonar nuestro proyecto hay que conectarse por SSH con nuestra cuenta de GitHub, esto lo lograremos con los siguientes pasos:

```
ssh-keygen -t rsa -b 4096 -C tucorreo@mail
```

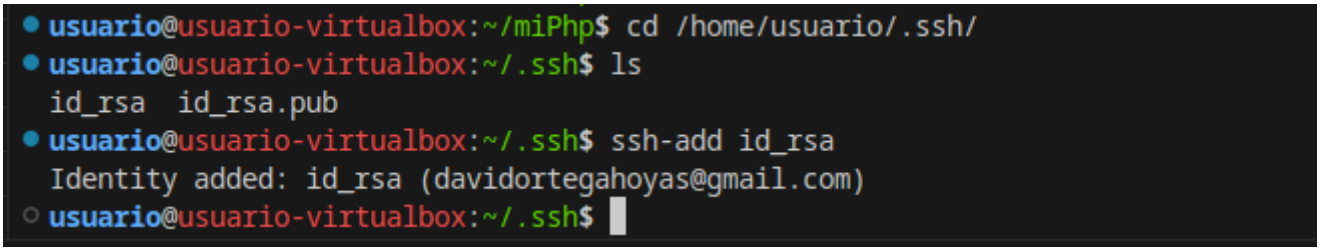
Ejecutamos el comando "ssh-keygen -t rsa -b 4096 -C tucorreo@mail" para generar la clave pública y la privada. Cuando nos pida donde guardarlo ponemos /home/usuario/.ssh/id_rsa . La passphrase la dejamos vacia dandole a enter.




```
cd /home/usuario/.ssh/
```

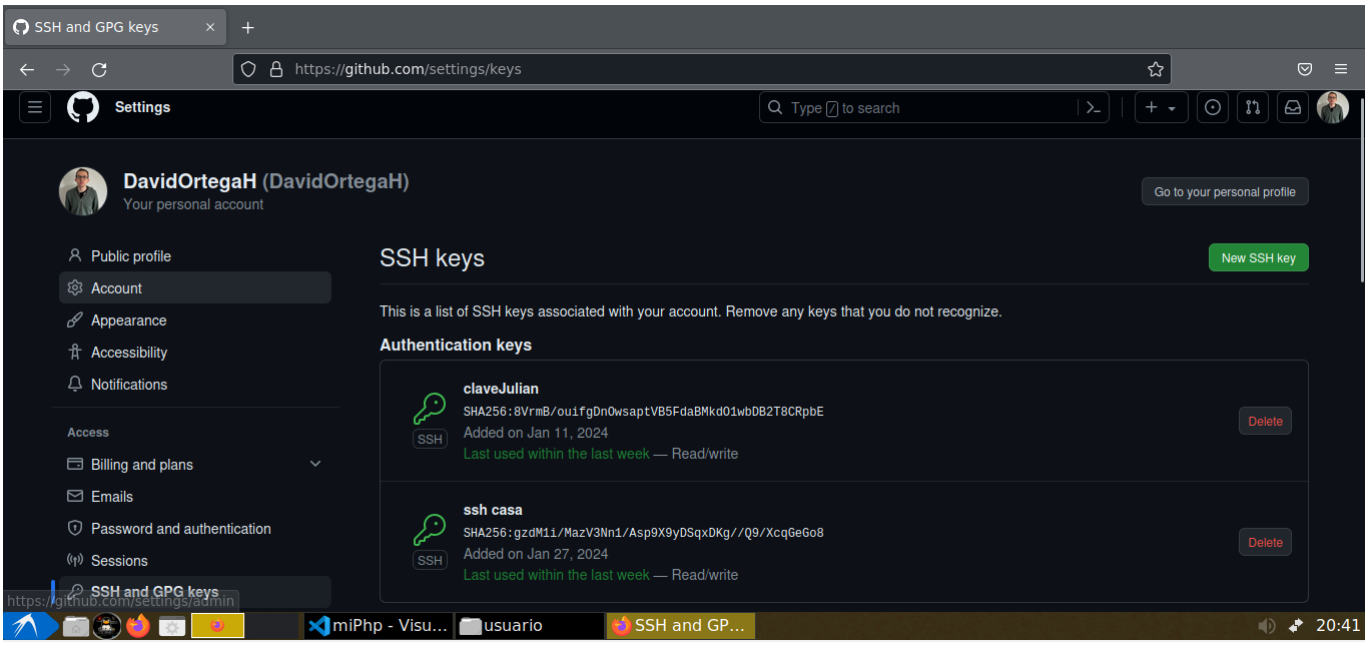
```
ssh-add id_rsa
```

Ahora nos movemos al directorio en cuestión con el comando "cd /home/usuario/.ssh/", despues ejecutamos "ssh-add id_rsa" para añadir la clave privada.

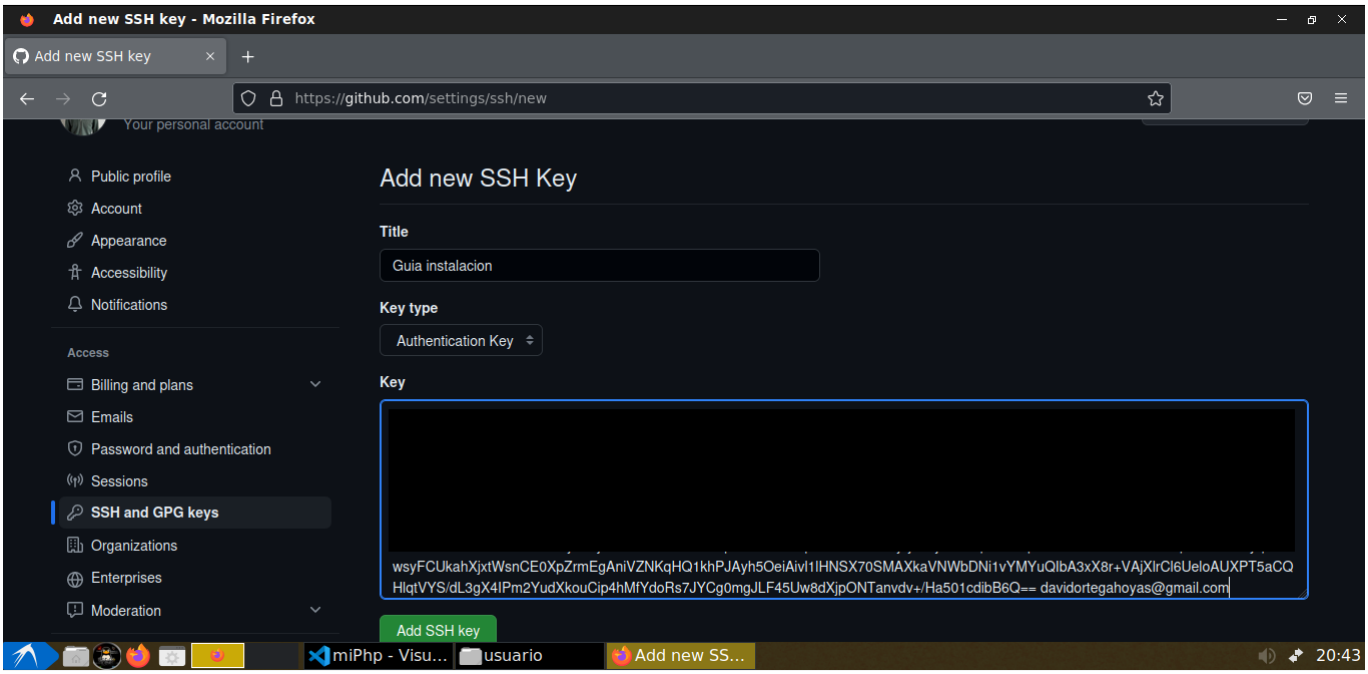


Luego hacemos "cat id_rsa.pub" y copiamos todo lo que nos devuelve.

Ahora nos logeamos en Github y vamos a https://github.com/settings/keys . Entonces pulsamos en New SSH Key.

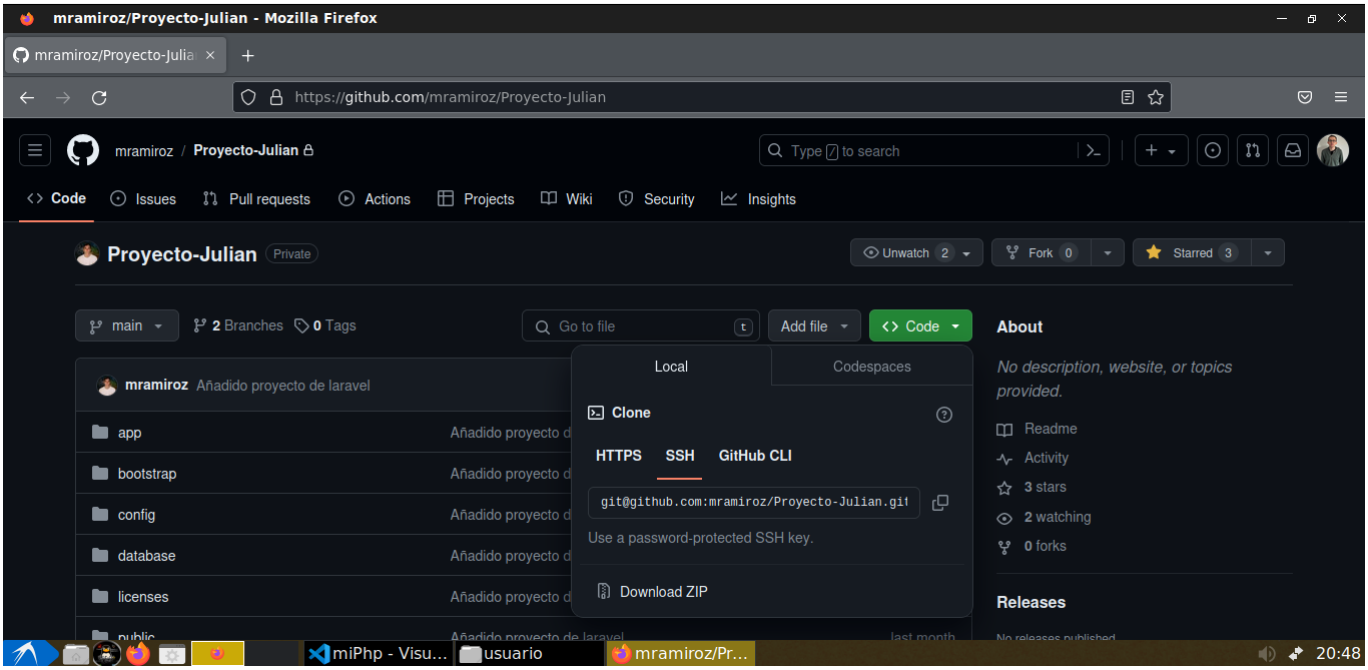


Una vez dentro le ponemos un título y en Key pegamos el tocho de la clave que copiamos antes. Entonces pulsamos en Add Key



3.2 Git clone

Nos vamos al repositorio de github del proyecto https://github.com/mramiroz/Proyecto-Julian y copiamos la opción SSH para clonarlo.




```
git clone git@github.com:mramiroz/Proyecto-Julian.git
```

En la carpeta miPHP hacemos un "git clone git@github.com:mramiroz/Proyecto-Julian.git", en la opción de que si queremos seguir conectando le decimos que yes.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

usuario@usuario-virtualbox:~/miPhp$ ls
usuario@usuario-virtualbox:~/miPhp$ git clone git@github.com:mramiroz/Proyecto-Julian.git
Clonando en 'Proyecto-Julian'...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 1340, done.
remote: Counting objects: 100% (74/74), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 1340 (delta 28), reused 52 (delta 21), pack-reused 1266
Recibiendo objetos: 100% (1340/1340), 5.61 MiB | 6.46 MiB/s, listo.
Resolviendo deltas: 100% (733/733), listo.
usuario@usuario-virtualbox:~/miPhp$
```

3.3 Evitar problemas de permisos

```
sudo chown -R usuario Proyecto-Julian/
```

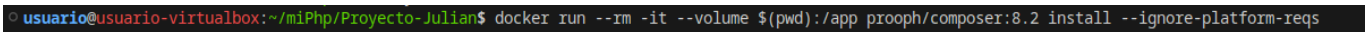
Para hacerlo ejecutamos el comando "sudo chown -R usuario Proyecto-Julian/"

```
usuario@usuario-virtualbox:~/miPhp$ sudo chown -R usuario Proyecto-Julian/
```

3.4 Generar el vendor

```
docker run --rm -it --volume $(pwd):/app prooph/composer:8.2 install --ignore-
platform-reqs
```

Para conseguir esta hazaña solo hay usar el comando "docker run --rm -it --volume $(pwd):/app prooph/composer:8.2 install --ignore-platform-reqs" en la carpeta del proyecto.

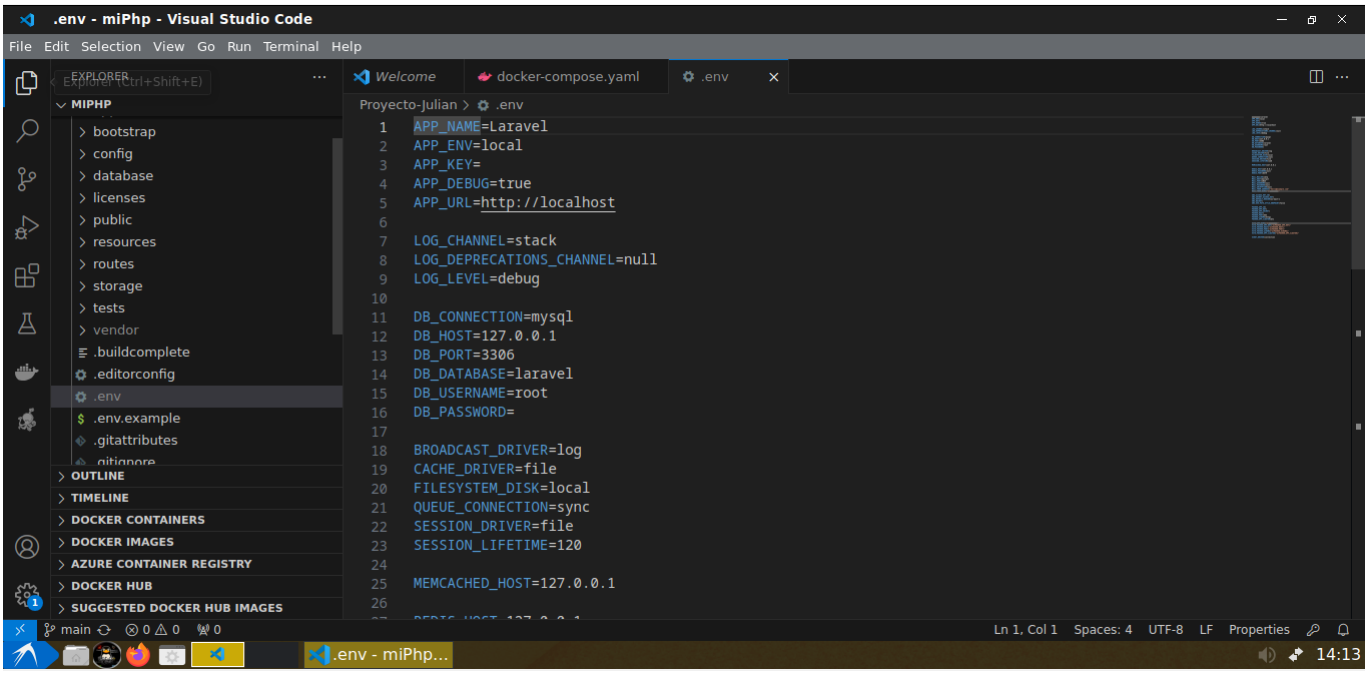


3.5 Crear el .env del proyecto

Este archivo es importante tenerlo en la carpeta del proyecto, debemos de crearlo, lo llamamos .env y dentro ponemos lo siguiente (si no lo copiamos del .env.example):

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailpit
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
```

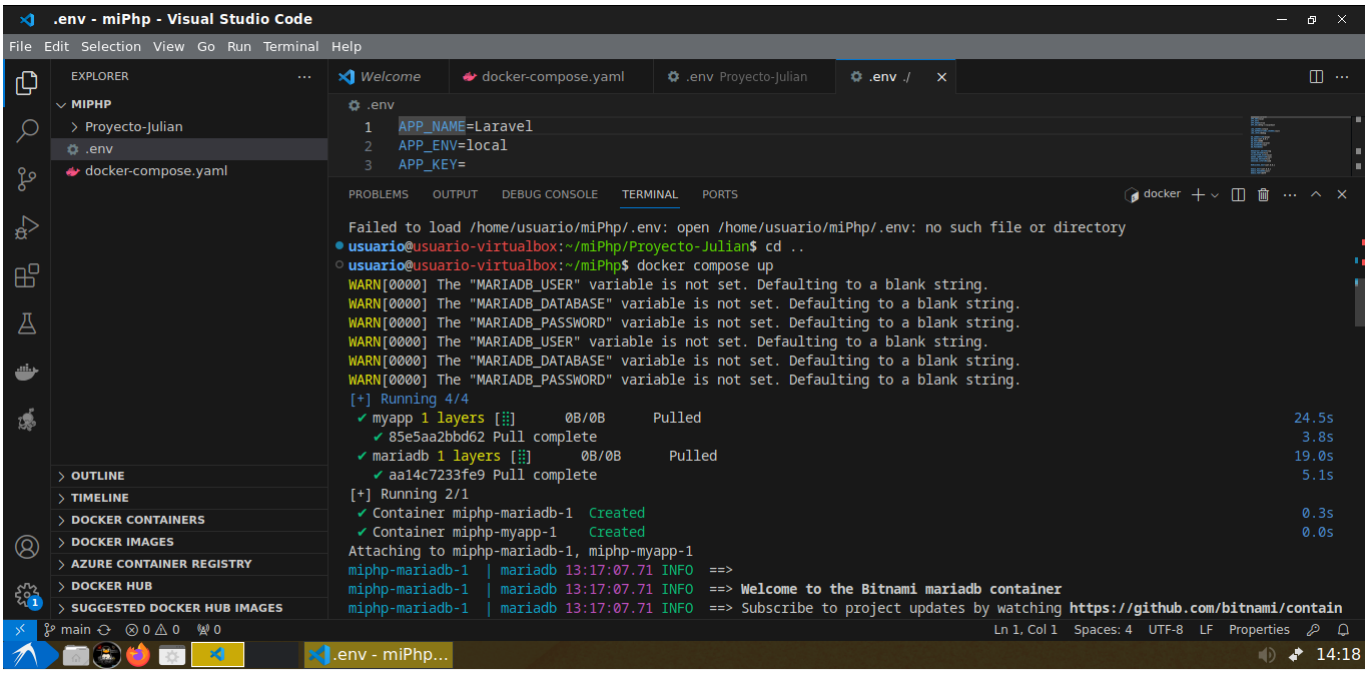
```
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1

VITE_APP_NAME="${APP_NAME}"
VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
VITE_PUSHER_HOST="${PUSHER_HOST}"
VITE_PUSHER_PORT="${PUSHER_PORT}"
VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

SCOUT_DRIVER=collection
```



3.6 Compose up

```
docker compose up
```

Una vez hecho esto tiramos el comando "docker compose up" desde la carpeta miphp.



3.7 Instalar npm

```
sudo apt-get install npm
```

Como nuestro proyecto utiliza Vite, hemos de instalar npm, hay que ejecutar "sudo apt-get install npm"

```
usuario@usuario-virtualbox:~/miPhp/Proyecto-Julian$ sudo apt-get install npm
```

```
sudo apt-get update
```

Además de "sudo apt-get update"

```
usuario@usuario-virtualbox:~/miPhp/Proyecto-Julian$ sudo apt-get update
```

3.8 Instalar Vite

```
npm install vite
```

Hacemos "npm install vite"

```
usuario@usuario-virtualbox:~/miPhp/Proyecto-Julian$ npm install vite
```

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo
sysctl -p
```

Ahora nos daría un error por eso tenemos que ejecutar el comando "echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p"

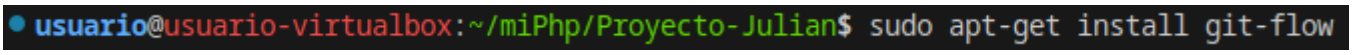

```
npm run dev
```

Con el error subsanado podemos ejecutar "npm run dev", de esta manera veremos nuestra aplicación funcionando en localhots:8000

3.9 Instalar git flow

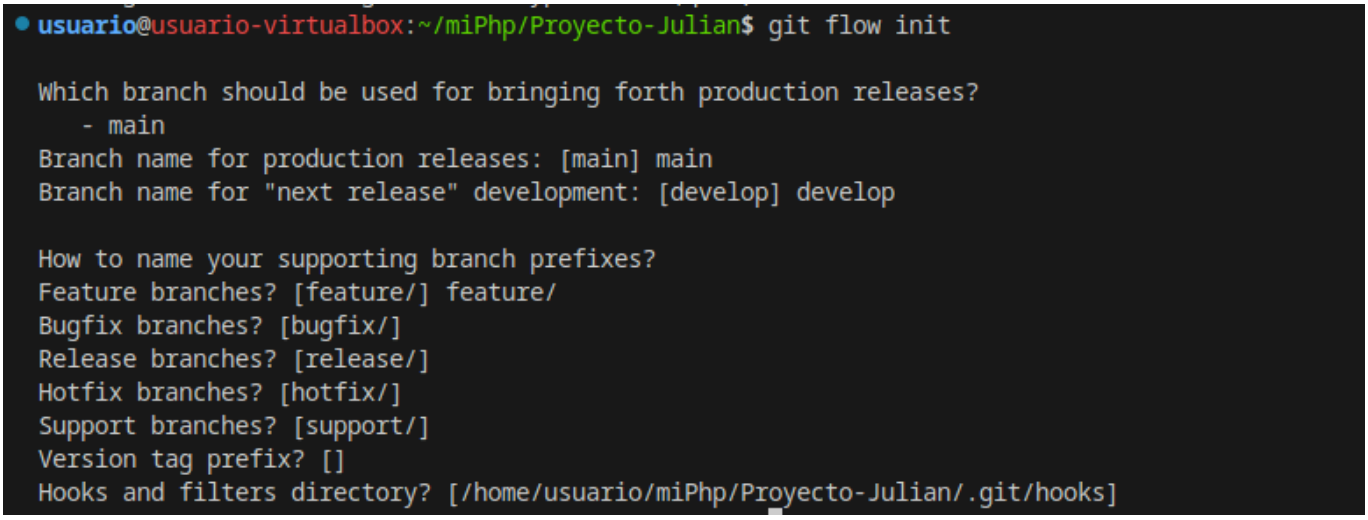
```
sudo apt-get install git-flow
```

Ya que usamos git flow toca instalarlo con "sudo apt-get install git-flow".




```
git flow init
```

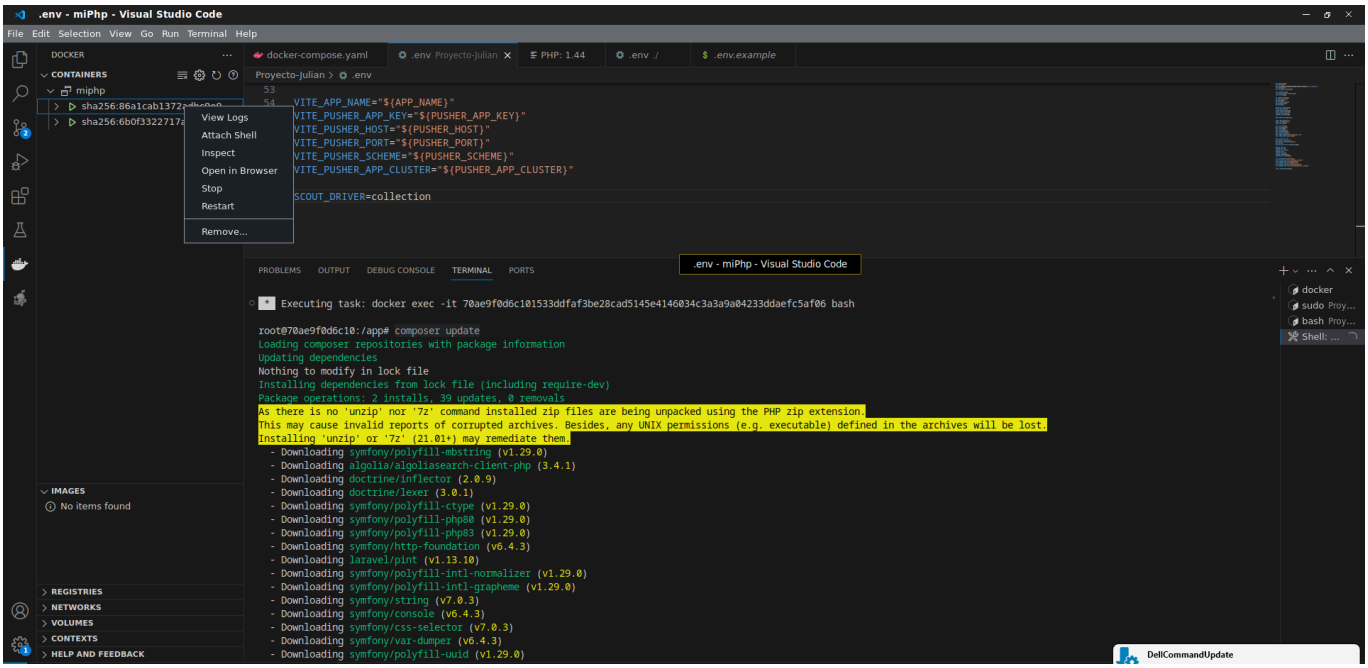
Luego hacemos "git flow init", respondemos como se ve en la imagen.



3.10 Como podrás observar la cosa sigue sin funcionar

```
composer update
```

Para arreglarlo nos metemos en la terminal del contenedor de docker y ejecutamos "composer update"



3.11 Problemas con git

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Tu Nombre"
```

Tirar estos comandos si hace falta con vuestro datos.

"git config --global user.email "you@example.com"" "git config --global user.name "Tu Nombre""

FIN