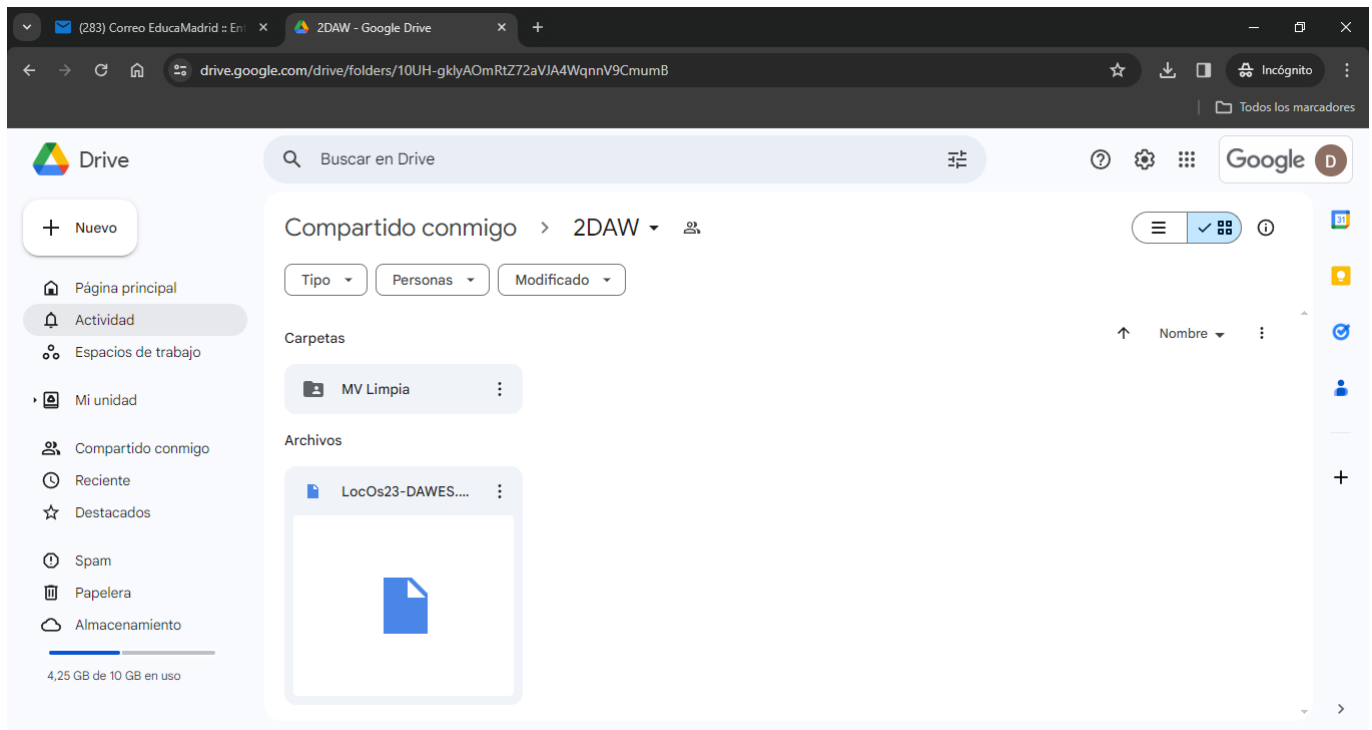
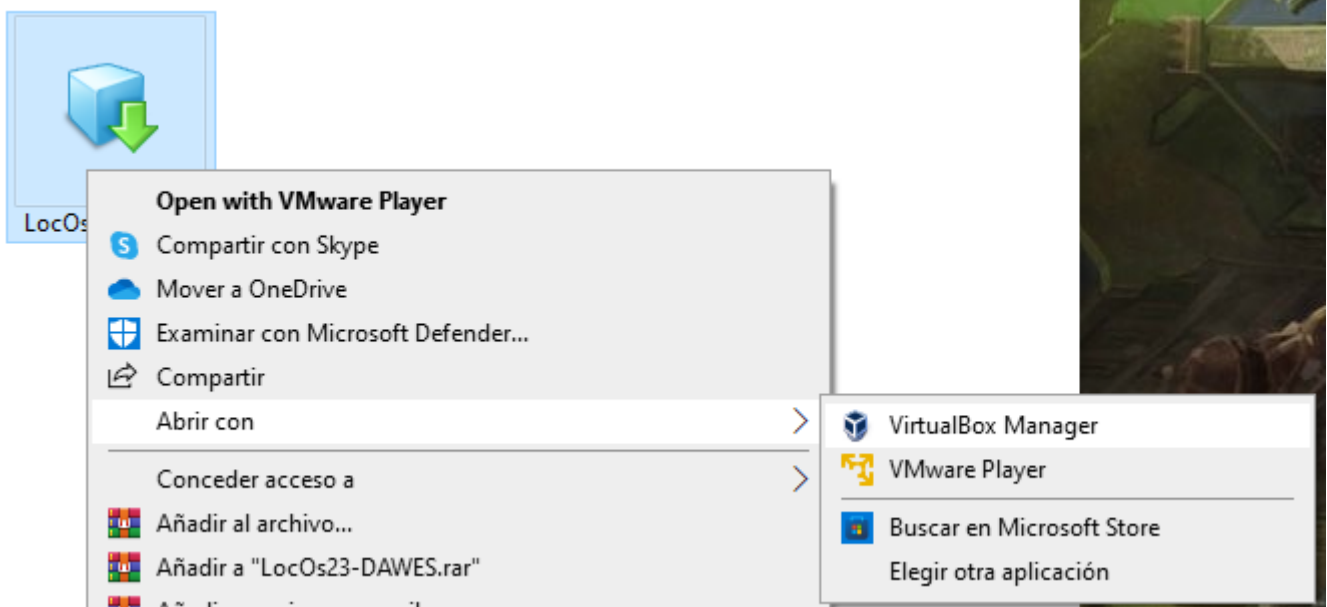


- Adquirir la ova de la máquina virtual: Para conseguir la OVA nos vamos al repositorio que corresponda, como ahora no estoy en el examen toca meterse en Google Drive con la cuenta de educa.madrid.org, una vez en drive seleccionamos a la izquierda "Compartido conmigo" -> "2DAW" y luego descargamos la 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"

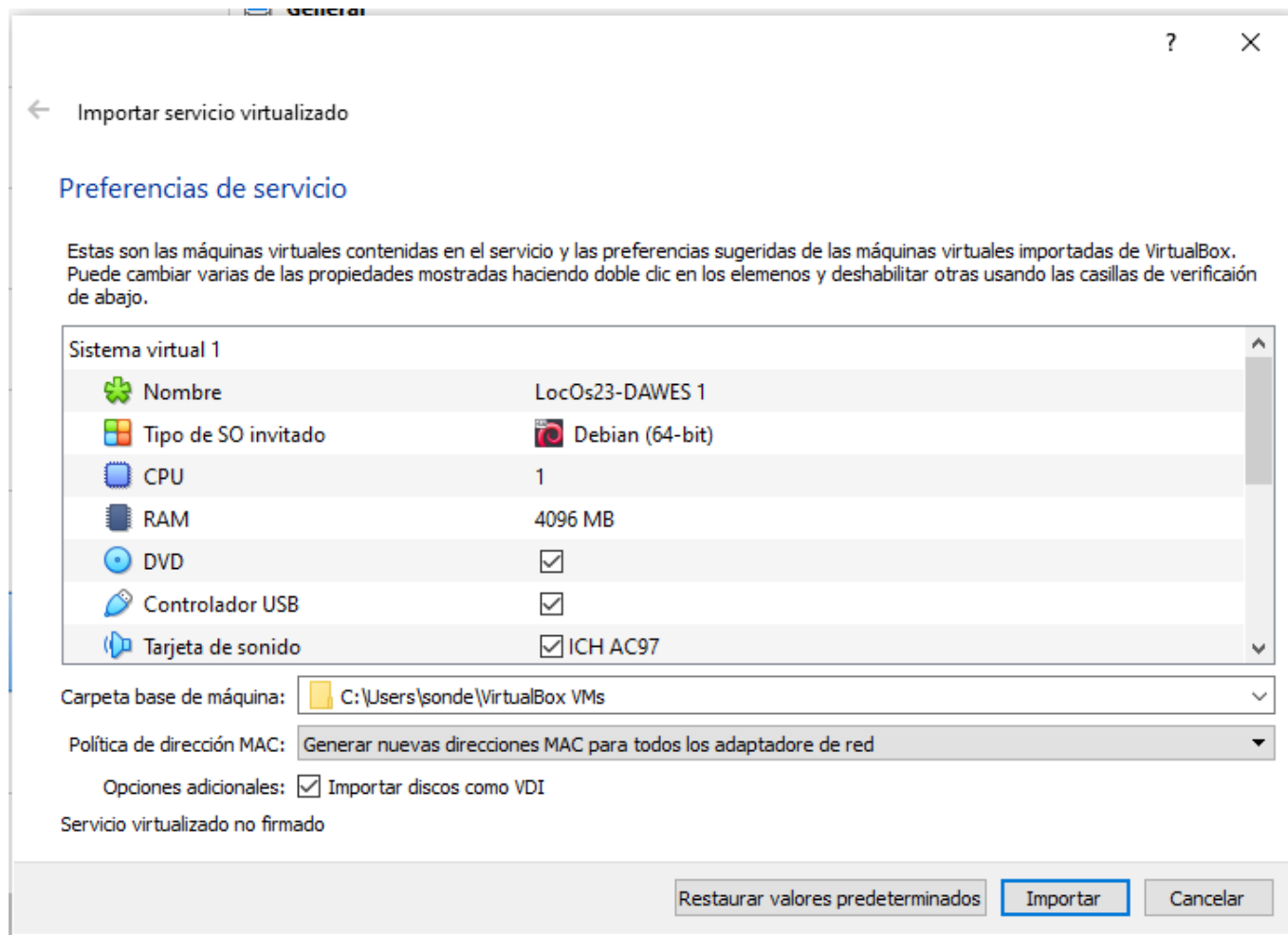


- Instalar la máquina virtual:

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

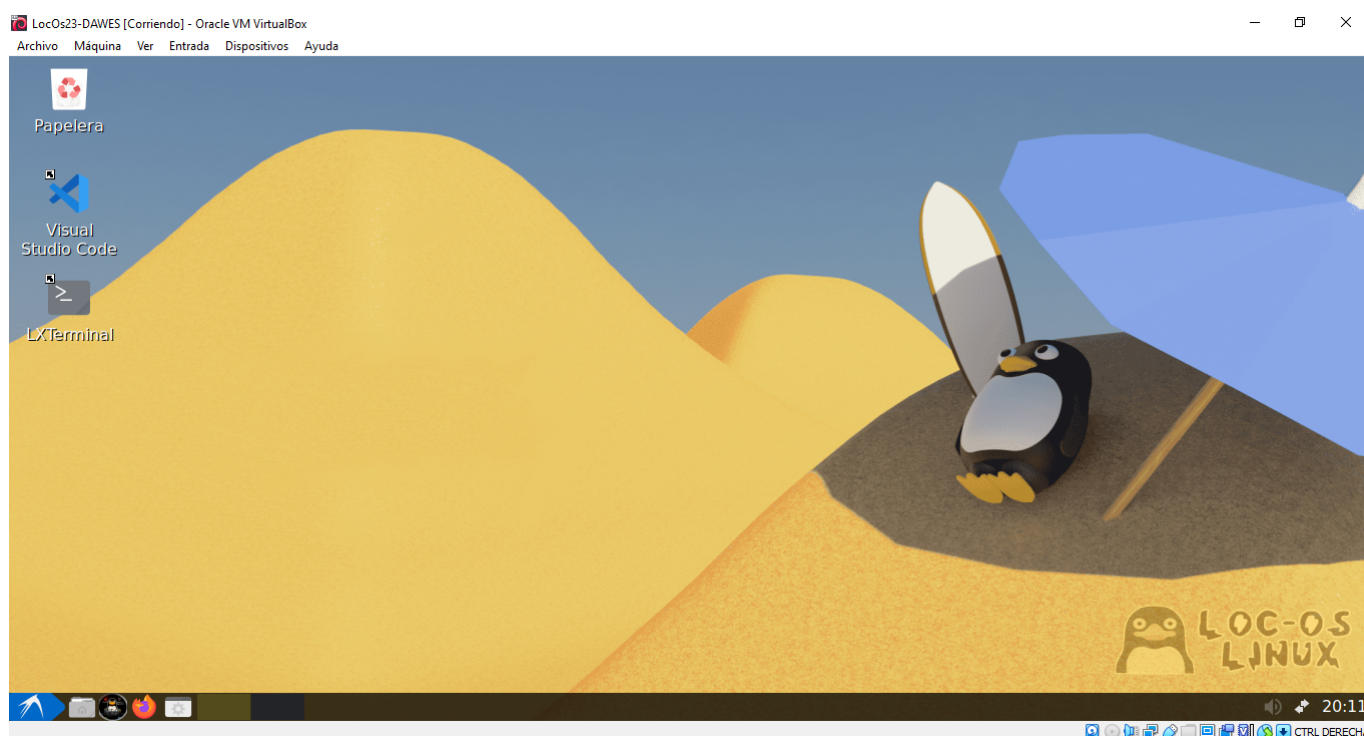


Os aparecera el menú de VirtualBox, dejais todo como está excepto que en "Política de direcciones MAC" abris el desplegable y elegís generar nuevas MAC para todos los adaptadores. Le dais a importar.



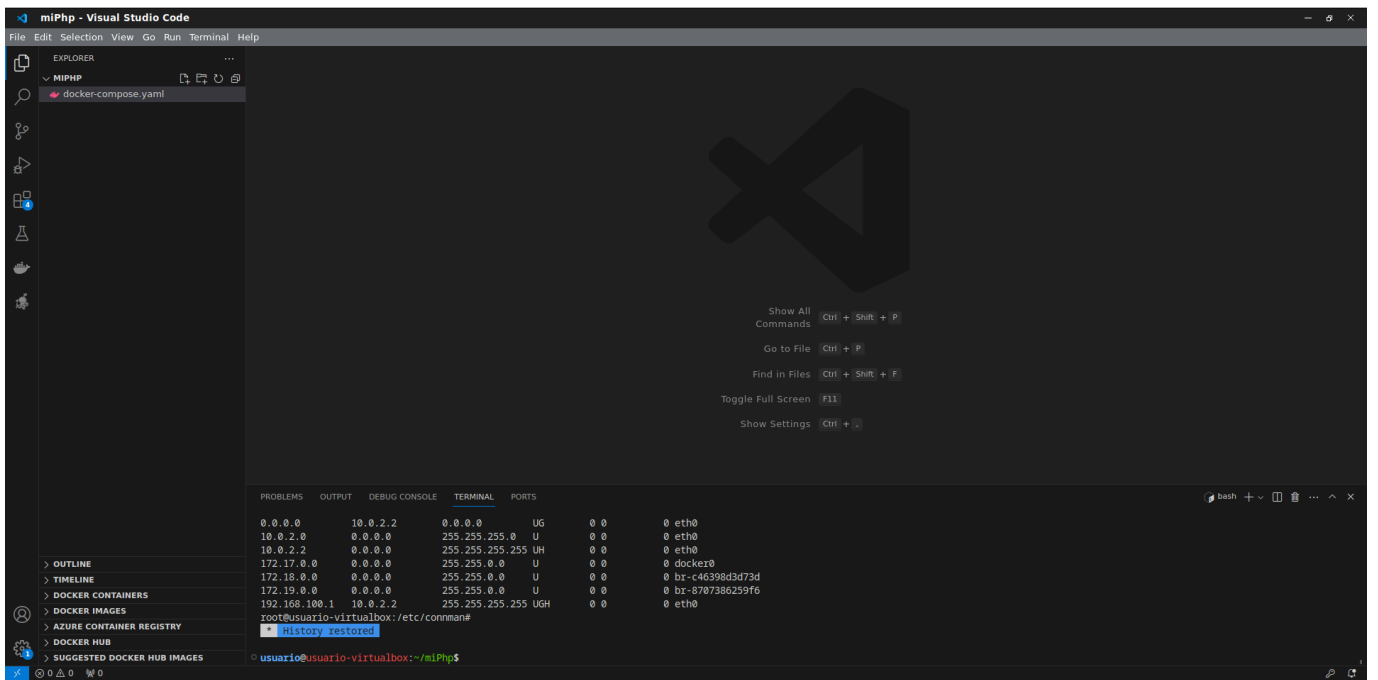
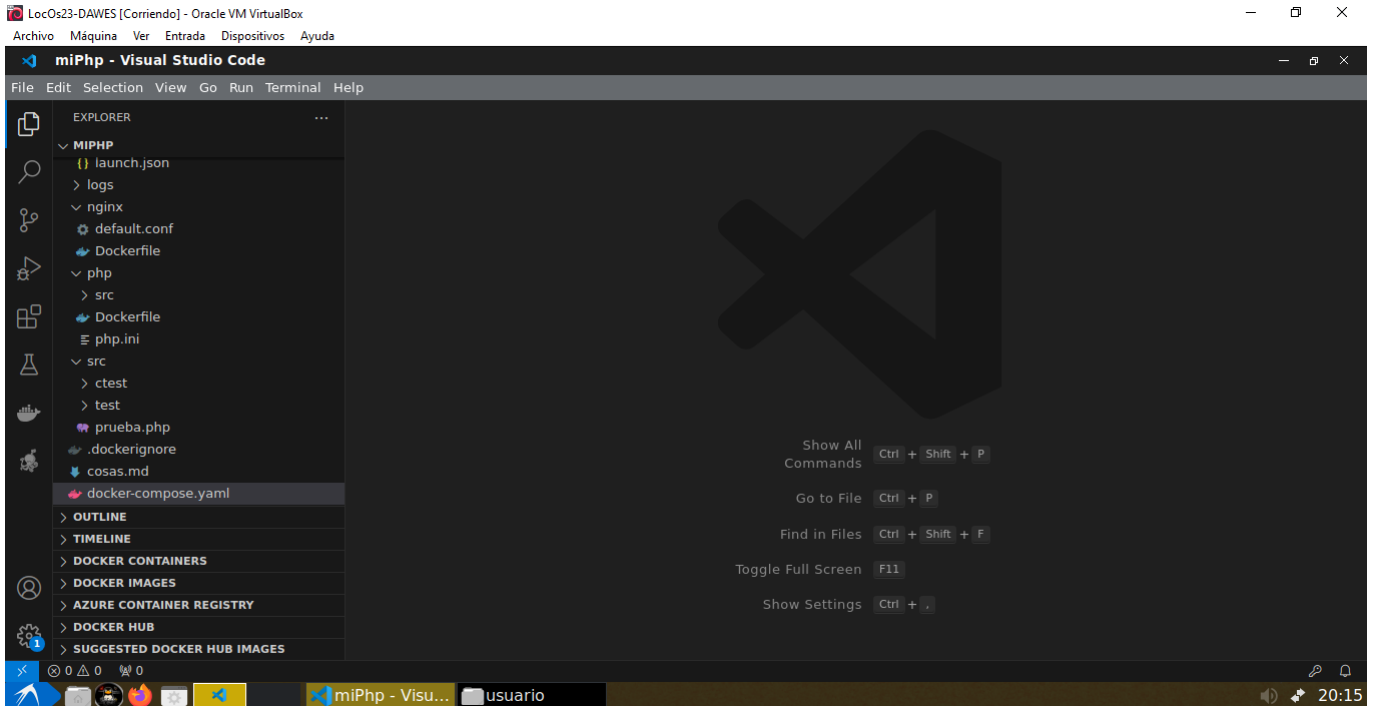
- Abrir la máquina virtual:

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

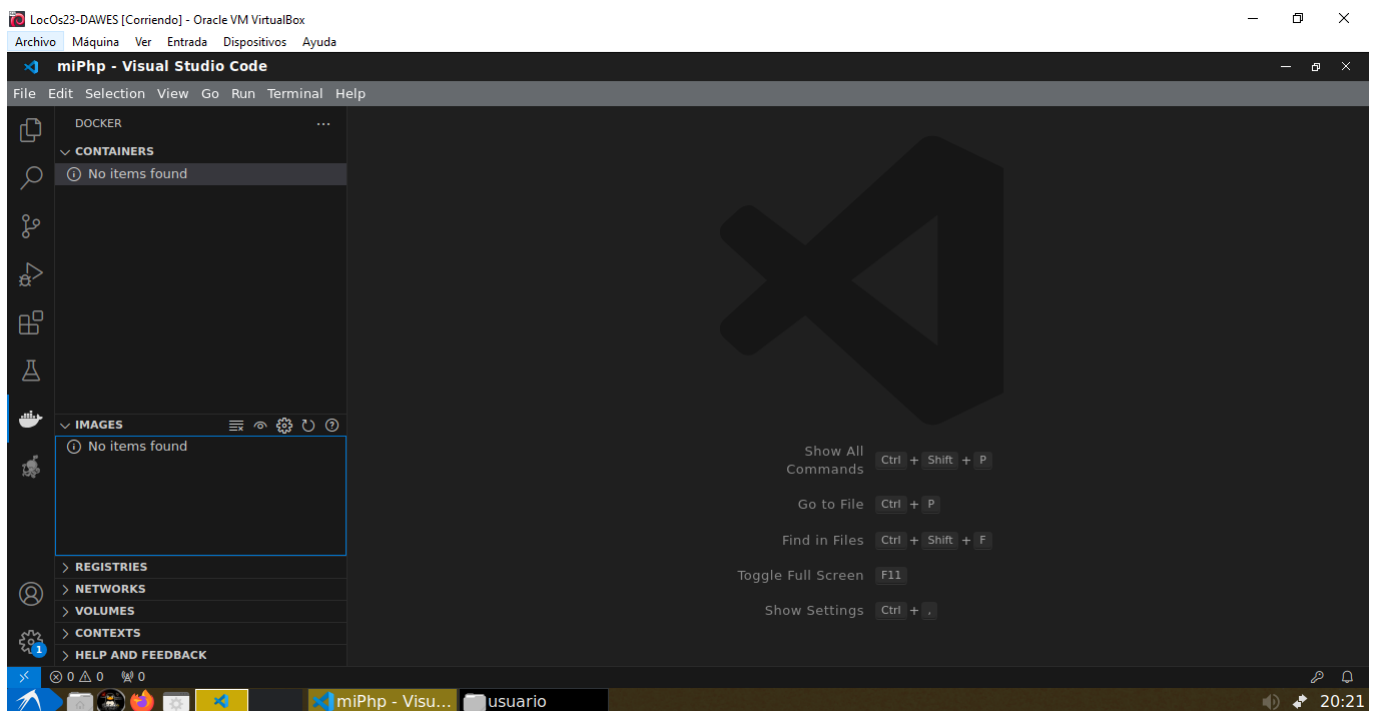
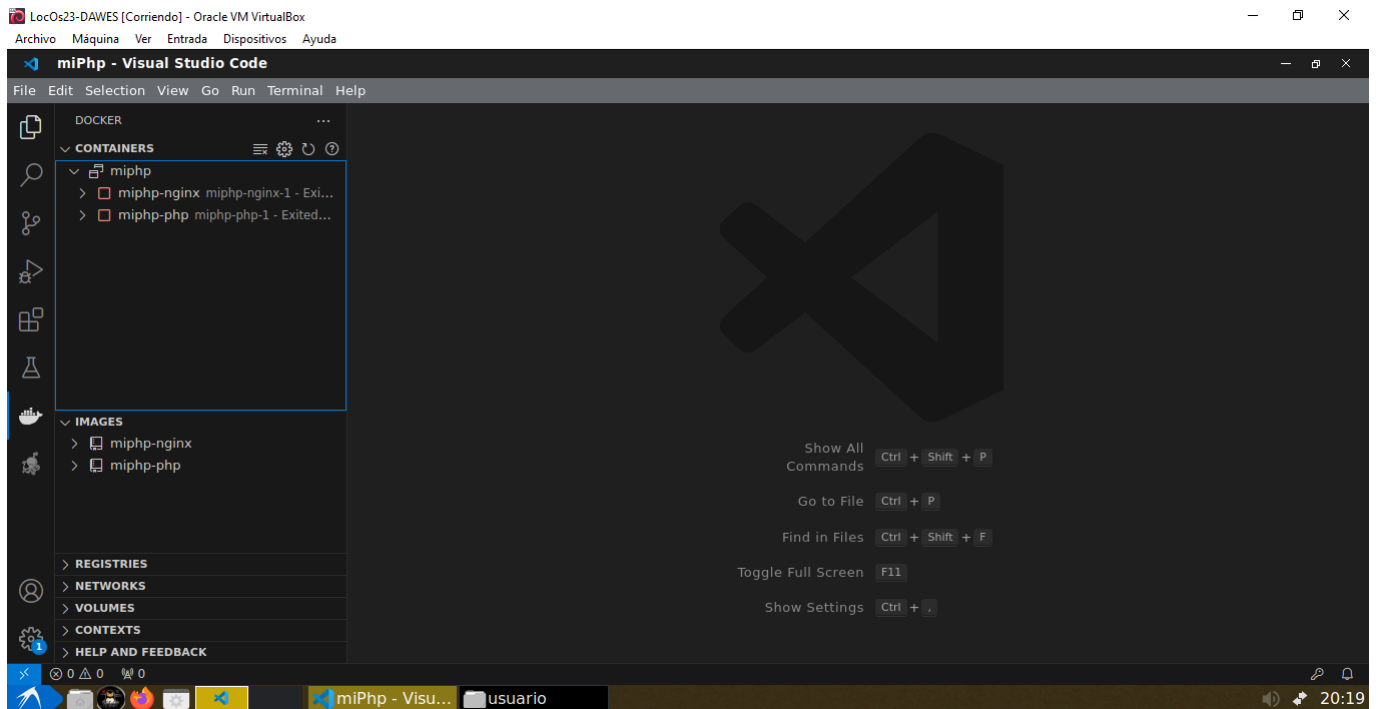


- Preparar docker:

Abrimos el visual y nos deberíamos 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
```

- Hacer el .env del docker

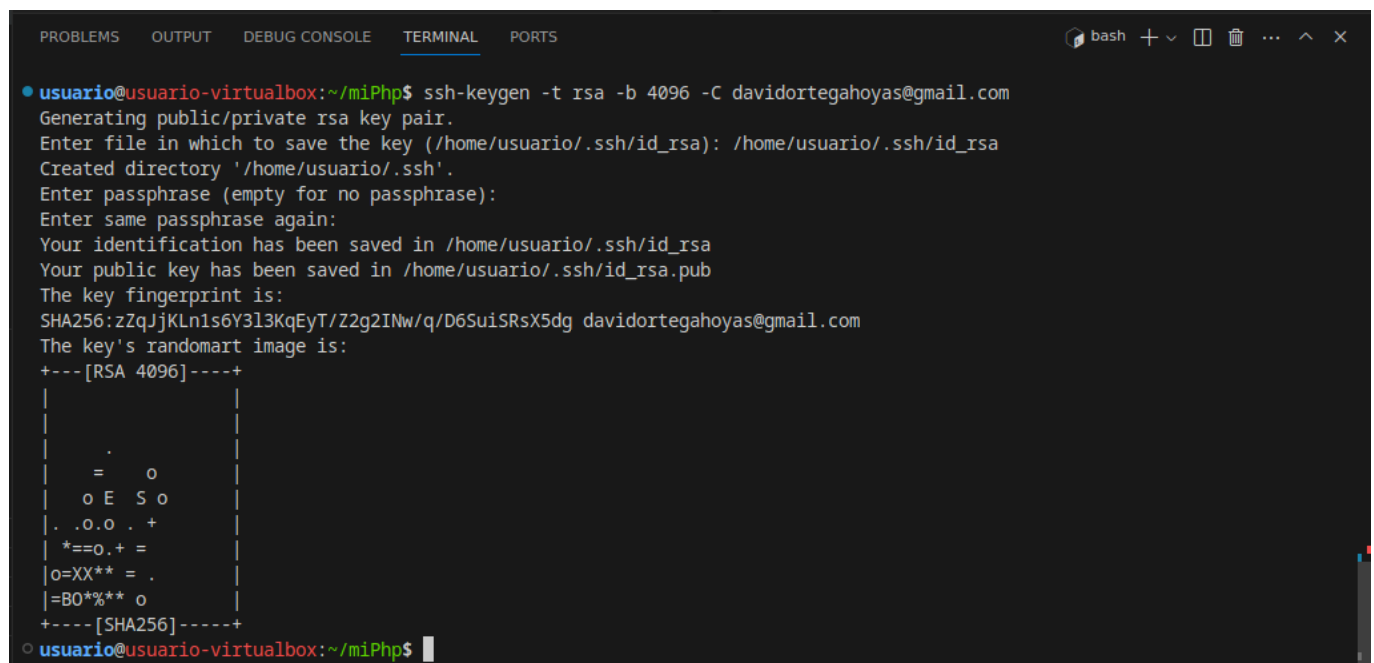
Crearemos en miPHP un archivo llamado ".env" con el siguiente contenido:

MaribaDB - Desarrollo

```
MARIADB_VERSION=8.0.21 MARIADB_HOST=mariadb MARIADB_DATABASE=bitnami_myapp
MARIADB_USER=bn_myapp MARIADB_PASSWORD=secret
```

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

** 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.

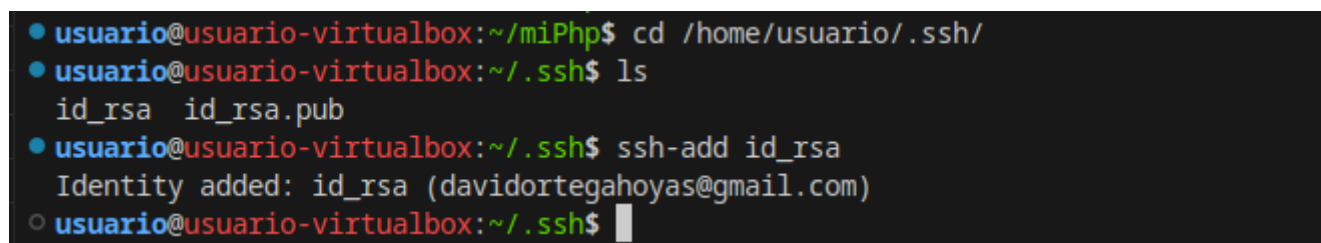


```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● usuario@usuario-virtualbox:~/miPhp$ ssh-keygen -t rsa -b 4096 -C davidortegahoyas@gmail.com
Generating public/private rsa key pair.
Enter file in which to save the key (/home/usuario/.ssh/id_rsa): /home/usuario/.ssh/id_rsa
Created directory '/home/usuario/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/usuario/.ssh/id_rsa
Your public key has been saved in /home/usuario/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:zZqJjKLn1s6Y3l3KqEyT/Z2g2INw/q/D6SuisRsX5dg davidortegahoyas@gmail.com
The key's randomart image is:
+---[RSA 4096]-----+
|
|      .
|      =   o
|    o E   S o
|   . . O . +
|  *==O.+ =
|O=XX** = .
|B0*** o
+---[SHA256]-----+
○ usuario@usuario-virtualbox:~/miPhp$

```

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.



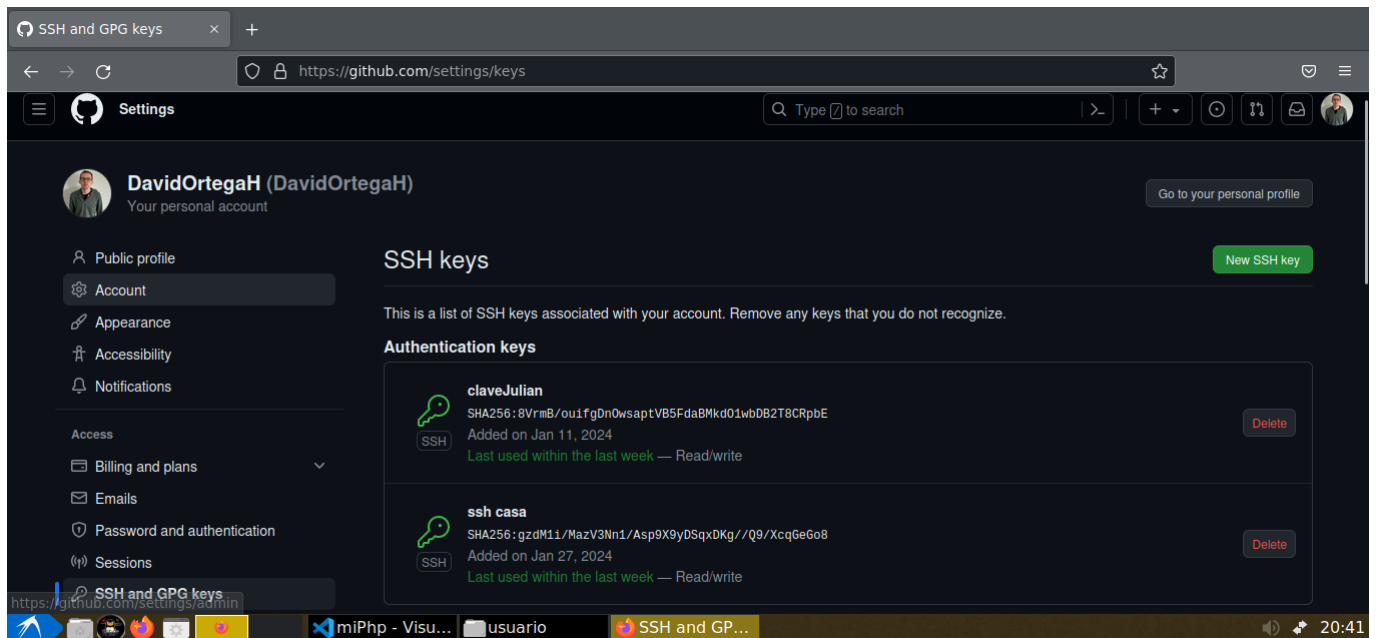
```

● usuario@usuario-virtualbox:~/miPhp$ cd /home/usuario/.ssh/
● usuario@usuario-virtualbox:~/.ssh$ ls
id_rsa  id_rsa.pub
● usuario@usuario-virtualbox:~/.ssh$ ssh-add id_rsa
Identity added: id_rsa (davidortegahoyas@gmail.com)
○ usuario@usuario-virtualbox:~/.ssh$

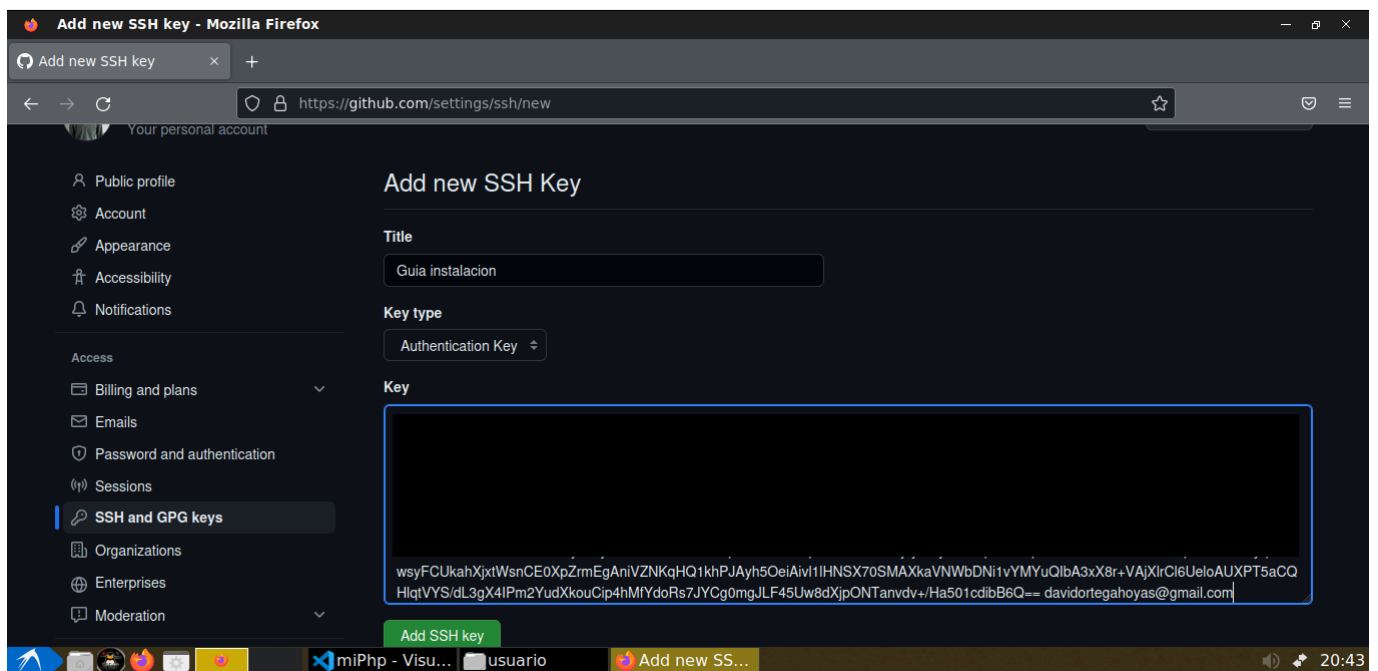
```

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

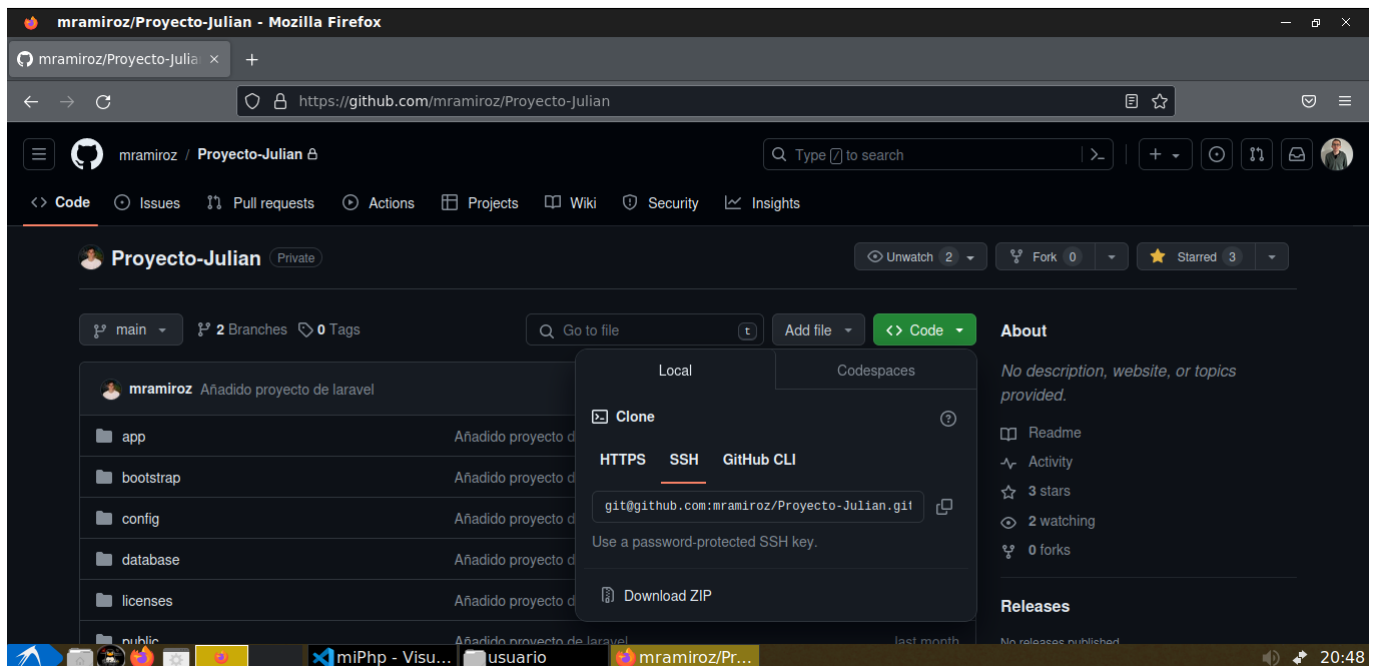
** Ahora nos logeamos en Github y vamos a <https://github.com/settings/keys> . Entonces pulsamos en New SSH Key.



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



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



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:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCQqU.
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$

```

- Evitar problemas de permisos: Para hacerlo ejecutamos el comando "sudo chown -R usuario Proyecto-Julian/"

```

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

```

- Generar el vendor: 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.

```

• usuario@usuario-virtualbox:~/miPhp/Proyecto-Julian$ docker run --rm -it --volume $(pwd):/app prooph/composer:8.2 install --ignore-platform-reqs

```

- 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

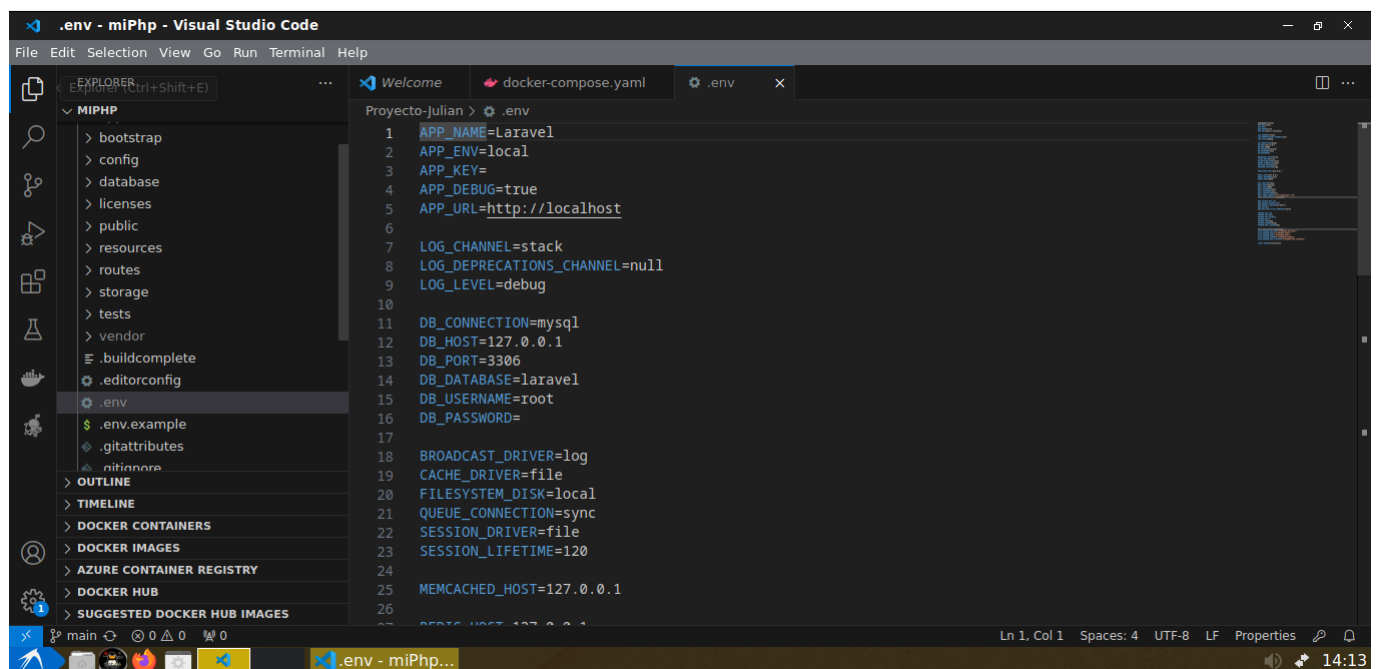
MAIL_MAILER=smtp MAIL_HOST=mailpit MAIL_PORT=1025 MAIL_USERNAME=null MAIL_PASSWORD=null
MAIL_ENCRYPTION=null MAIL_FROM_ADDRESS="hello@example.com" MAIL_FROM_NAME="\${APP_NAME}"

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}"

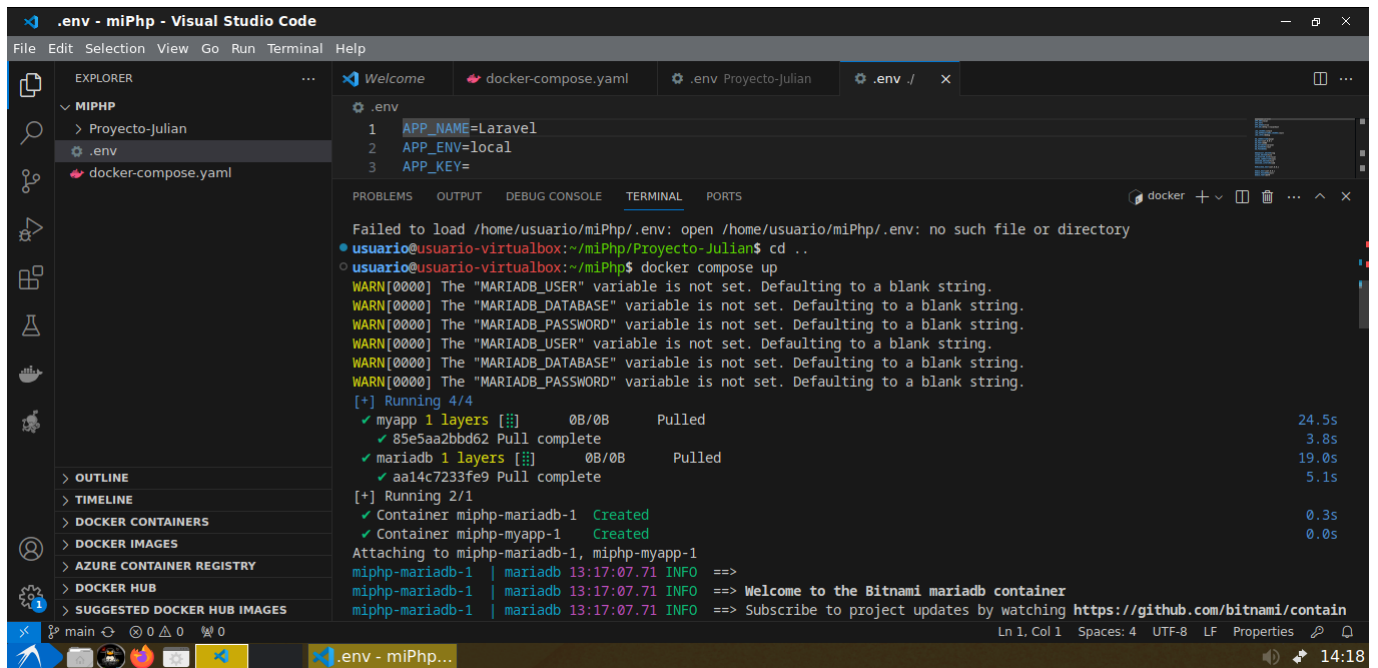
SCOUT_DRIVER=collection



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
```

- Compose up:

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



- Instalar 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
```

Además de "sudo apt-get update"

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

- Instalar Vite Hacemos "npm install vite"

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

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"

```
usuario@usuario-virtualbox:~/miPhp/Proyecto-Julian$ echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p
```

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

- Instalar git flow

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

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

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

```

• usuario@usuario-virtualbox:~/miPhp/Proyecto-Julian$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main] main
Branch name for "next release" development: [develop] develop

How to name your supporting branch prefixes?
Feature branches? [feature/] feature/
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/usuario/miPhp/Proyecto-Julian/.git/hooks]

```

- Como podrás observar la cosa sigue sin funcionar.

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

```

.env - miPhp - Visual Studio Code
File Edit Selection View Go Run Terminal Help
DOCKERS
CONTAINERS
miPhp
sha256:86a1cab1372...
sha256:6b0f3322717...
View Logs
Attach Shell
Inspect
Open in Browser
Stop
Restart
Remove...
docker-compose.yaml
.env Proyecto-Julian
PHP: 1.44
.env /
.env.example
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
.env - miPhp - Visual Studio Code
Executing task: docker exec -it 70ae9f0d6c101533ddaf3be28cad5145e4146834c3a3a9a04233ddaefc5af06 bash
root@70ae9f0d6c10:/app# composer update
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 39 updates, 0 removals
As there is no 'unzip' nor '7z' command installed zip files are being unpacked using the PHP zip extension.
This may cause invalid reports of corrupted archives. Besides, any UNIX permissions (e.g. executable) defined in the archives will be lost.
Installing 'unzip' or '7z' (21.01+) may remediate them.
- Downloading symfony/polyfill-mbstring (v1.29.0)
- Downloading algolia/algoliasearch-client-php (3.4.1)
- Downloading doctrine/inferno (2.0.9)
- Downloading doctrine/lexer (3.0.1)
- Downloading symfony/polyfill-ctype (v1.29.0)
- Downloading symfony/polyfill-php80 (v1.29.0)
- Downloading symfony/polyfill-php83 (v1.29.0)
- Downloading symfony/http-foundation (v6.4.3)
- Downloading laravel/pint (v1.13.10)
- Downloading symfony/polyfill-intl-normalizer (v1.29.0)
- Downloading symfony/polyfill-intl-grapheme (v1.29.0)
- Downloading symfony/string (v7.0.3)
- Downloading symfony/console (v6.4.3)
- Downloading symfony/css-selector (v7.0.3)
- Downloading symfony/var-dumper (v6.4.3)
- Downloading symfony/polyfill-uuid (v1.29.0)

```

- Eso es todo.

A esta guía le falta actualizarse.