



# PROGRAMA MODULAR EN ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

PRÁCTICO

## Proyecto

Título: Despliegue de infraestructura TI en contenedores docker y automatización de tareas siguiendo prácticas DevOps

Autor/a: Manuel Olmo Merino

Tutor/a: Elena Ruiz Larrocha

Curso académico: 2023/24



## AGRADECIMIENTOS

A Mamen, mi mujer, por su apoyo en el momento que decidí volver a estudiar -a mis años- para sacar el grado de ASIR. Sin su ánimo y su infinita paciencia durante este tiempo no habría llegado hasta aquí.

## Índice de contenido

1.	Resumen/Summary.....	6
2.	Objetivos generales y específicos y alcance del proyecto .....	8
3.	Definiciones.....	10
	DevOps .....	10
	VirtualBox.....	10
	Docker .....	10
	Docker Compose .....	10
	Docker Hub.....	10
	Ansible.....	10
	Visual Studio Code.....	10
	PHP .....	10
4.	Notaciones y símbolos .....	11
5.	Preparación del entorno de trabajo.....	12
	5.1. Despliegue de máquina virtual Ubuntu .....	12
	5.2. Instalación de Docker .....	12
	5.3. Instalación de Ansible .....	16
	5.4. Instalación Visual Studio Code .....	18
6.	Operaciones. Orquestación Docker Compose .....	21
	6.1. Definición de la red Docker .....	22
	6.2. Definición de los servicios .....	22
	6.2.1. Ficheros Dockerfile.....	25
	6.3. Ejecución de la orquestación Docker Compose .....	26
	6.4. Repositorio de Docker Hub .....	30
	6.5. Eliminación y recreación de la orquestación .....	33
	6.6. Recreación del entorno en otro sistema.....	34
7.	Desarrollo. Frontend HTML, PHP .....	39
	7.1. Frontend. Index.html .....	39
	7.2. Frontend. guardar_facturas.php.....	41
8.	Automatización de tareas con Ansible.....	43
	8.1. Generar claves SSH de los contenedores gestionados.....	43
	8.2. Creación de fichero de inventario y ansible.cfg .....	44
	8.3. Creación de tareas en Ansible (Playbook).....	45

9.	Conclusiones y recomendaciones .....	47
9.1.	Líneas futuras y recomendaciones.....	48
10.	Referencias Bibliográficas .....	50
10.1.	Cursos realizados.....	50

## Índice de figuras

Figura 1 Orquestación Docker Compose (Fuente: propia) .....	6
Figura 2 Notaciones y Símbolos (Fuente: propia) .....	11
Figura 3 MV Ubuntu (Fuente: propia) .....	12
Figura 4 Arquitectura Docker (Fuente: Curso Docker Apasoft).....	13
Figura 5 Componentes Docker (Fuente: Curso Docker Apasoft).....	14
Figura 6 Imágenes vs Contenedores (Fuente: Curso DevOps total).....	14
Figura 7 Componentes de Ansible(Fuente: Curso Ansible Apasoft) .....	17
Figura 8 Visual Studio Code (Fuente: propia) .....	19
Figura 9 Visual Studio Code extensión Docker (Fuente: propia) .....	19
Figura 10 Visual Studio Code extensión Ansible (Fuente :propia) .....	20
Figura 11 Entorno contenedores y servicios (Fuente: propia) .....	21
Figura 12 Frontend (Fuente: propia) .....	28
Figura 13 Acceso phpMyAdmin (Fuente: propia) .....	29
Figura 14 Datos formulario (Fuente: propia).....	29
Figura 15 Verificación datos phpMyAdmin (Fuente: propia).....	30
Figura 16 Repositorio Docker Hub (Fuente: propia) .....	32
Figura 17 Docker Hub versiones apache (Fuente: propia).....	33
Figura 18 Docker Hub descarga imagen MySQL (Fuente: propia).....	35
Figura 19 Verificación servicios en nuevo servidor (Fuente: propia) .....	37

## 1. Resumen/Summary

### Resumen

El objetivo del proyecto fin de grado es automatizar el despliegue y la configuración de un entorno de microservicios de red en un servidor Ubuntu usando contenedores Docker y gestionar el mantenimiento de estos servicios mediante Ansible. La implementación se basa en las mejores prácticas DevOps, lo que garantiza la eficiencia en el desarrollo, despliegue y mantenimiento de la aplicación. Esto incluye la automatización en el proceso de puesta en producción de los servicios necesarios.

El proyecto se centra en el despliegue de un entorno de microservicios para poner en marcha una aplicación stack. Esta aplicación requiere de varios componentes tecnológicos, incluyendo un servidor web Apache, una base de datos MySQL y un cliente gráfico Phpmyadmin. La orquestación de estos servicios se realiza mediante Docker Compose, lo que facilita la gestión y la escalabilidad del entorno.

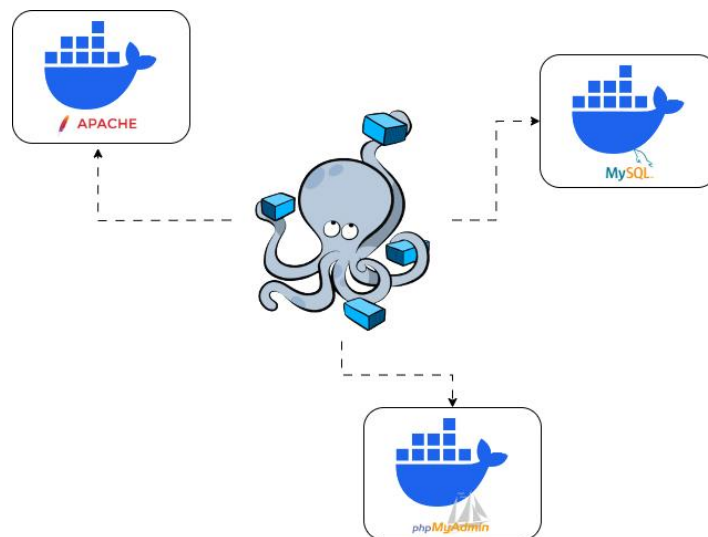


Figura 1 Orquestación Docker Compose (Fuente: propia)

Se espera obtener un entorno de microservicios completamente funcional y automatizado, capaz de desplegar y mantener una aplicación stack de manera eficiente y segura.

### Summary

The objective of the final degree project is to automate the deployment and configuration of a network microservices environment on an Ubuntu server using Docker containers and manage the maintenance of these services using Ansible. The implementation is based on DevOps best practices, ensuring efficiency in application development, deployment, and maintenance. This includes automation in the process of putting the necessary services into production.

The project focuses on the deployment of a microservices environment to launch a stack application. This application requires several technological components, including an Apache

web server, a MySQL database, and a Phpmyadmin graphical client. The orchestration of these services is done using Docker Compose, which facilitates the management and scalability of the environment.

It is expected to obtain a fully functional and automated microservices environment, capable of deploying and maintaining an application stack efficiently and securely.

## 2. Objetivos generales y específicos y alcance del proyecto

El objetivo principal del proyecto es el desarrollo y despliegue de los microservicios necesarios para implementar una sencilla aplicación web de facturación, utilizando tecnología de contenedores Docker dentro de un servidor con el sistema operativo Ubuntu 23.10 virtualizado sobre VirtualBox. Los microservicios a desplegar serán.

- Un frontend con HTML y PHP para la creación del formulario HTML y su conexión con la BBDD.
- Un backend con MySQL donde se creará la BBDD y al que se conectará el frontend.
- Un entorno gráfico para facilitar la gestión de la BBDD. Phpmyadmin.

Para ello se ha seguido la siguiente metodología.

- **Análisis de requisitos:** Identificación de los requisitos de la aplicación stack, así como de las herramientas y tecnologías necesarias para su despliegue y mantenimiento.
- **Diseño del sistema:** Definición de la arquitectura del sistema, incluyendo la estructura de contenedores Docker, red IP, volúmenes, software necesario para la integración de Ansible en el mantenimiento y automatización de tareas en los contenedores a desplegar.
- **Implementación:** Desarrollo e implementación de los scripts y configuraciones necesarias para desplegar y mantener el entorno de microservicios.

Para desplegar los microservicios se tendrán que realizar las siguientes tareas.

### 1. Configuración del entorno de desarrollo:

- a. Instalación del software de virtualización VirtualBox.
- b. Crear una máquina virtual con Ubuntu 23.10 en VirtualBox.
- c. Instalación de Docker en la máquina virtual Ubuntu 23.10
- d. Instalación de Ansible en la máquina virtual Ubuntu 23.10
- e. Instalación de Visual Studio Code en la máquina virtual Ubuntu 23.10

### 2. Desarrollo y orquestación de los microservicios con Docker Compose:

- a. Utilización de Docker Compose para definir la estructura de los servicios y su configuración
  - i. Definición de la red Docker.
  - ii. Definición del frontend a partir de una imagen phpapache de Docker Hub.
  - iii. Definición del backend a partir de una imagen MySQL de Docker Hub.
  - iv. Definición del entorno gráfico a partir de una imagen de Phpmyadmin de Docker Hub.
  - v. Creación de los ficheros de definición de imágenes Dockerfile para los microservicios de frontend y Phpmyadmin con el fin de instalar el servidor openssh y Python para la automatización de tareas a través de Ansible.



3. **Despliegue de la aplicación utilizando Docker Compose y verificar su funcionamiento.**
  - a. Probar el correcto despliegue de los microservicios y la interacción entre ellos.
4. **Desarrollo de los microservicios de frontend en HTML y PHP.**
5. **Automatización con Ansible.**
  - a. Definición de tareas en Ansible mediante Playbooks para la actualización e instalación de paquetes necesarios para el mantenimiento de los microservicios.

Cada microservicio se desplegará en un contenedor docker y estos contenedores estarán orquestados por Docker Compose, realizándose tareas de update e instalación de los paquetes necesarios para el posterior mantenimiento a través de Ansible al ejecutar la orquestación.

El objetivo del proyecto es **demostrar las ventajas de la automatización y optimización en la entrega de software propias de la metodología DevOps**, utilizando las herramientas más comunes en entornos DevOps hoy en día. La utilización de estas herramientas y el seguimiento de la metodología DevOps permite eliminar y recrear una orquestación de servicios y, por tanto, desplegar de nuevo la aplicación stack de manera eficiente y rápida con un solo comando y en menos de 5 minutos.

### 3. Definiciones

#### DevOps

El término DevOps surge de la combinación de dos términos: Desarrollo y Operaciones. Es una nueva cultura y metodología de trabajo que implica una mejor unión y colaboración entre el equipo de desarrollo de software y el equipo de operaciones, incrementando el proceso de automatización en el desarrollo de software.

#### VirtualBox

Software de virtualización gratuito de código abierto que permite la creación y ejecución de máquinas virtuales.

#### Docker

Docker es un proyecto de código abierto que permite automatizar el despliegue de aplicaciones dentro de contenedores.

#### Docker Compose

Docker Compose es una herramienta para definir y orquestar aplicaciones multi-contenedor, lo que facilita la gestión del stack de aplicaciones y mejora el desarrollo y despliegue de estas. Incluye comandos para la administración del ciclo de vida de la aplicación permitiendo iniciar, detener y reconstruir servicios de manera sencilla.

#### Docker Hub

Docker Hub es un servicio en la nube proporcionado por Docker, Inc. que sirve como repositorio público y privado para contenedores Docker. Es útil para que los desarrolladores y contribuyentes de código abierto encuentren, utilicen y compartan sus imágenes de contenedores actuando como una biblioteca de imágenes.

#### Ansible

Ansible es una aplicación open source que permite la gestión de la infraestructura TI automatizando las tareas más comunes en cualquier sistema informático, como el despliegue de aplicaciones, la gestión y configuración, el aprovisionamiento y la automatización de la infraestructura.

#### Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web.

#### PHP

PHP es un lenguaje muy popular para la creación de páginas web desde el lado del servidor, cuya sintaxis está basada en C y Perl. Sus ventajas principales es que es un lenguaje multiplataforma, con licencia GNU con una gran comunidad de usuarios lo que hace que tenga una red extensa de consultas.

## 4. Notaciones y símbolos

En el proyecto las órdenes se ejecutarán en la terminal del servidor Ubuntu, apareciendo sobre fondo negro con letras en blanco y alineado a la izquierda como se puede observar en el ejemplo.

```
root@ubuntuserv:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 23.10
Release:        23.10
Codename:       mantic
```

Figura 2 Notaciones y Símbolos (Fuente: propia)

## 5. Preparación del entorno de trabajo

En este capítulo se va a desplegar la máquina virtual Ubuntu en la que se instalará el software necesario para la ejecución del proyecto.

### 5.1. Despliegue de máquina virtual Ubuntu

Para el desarrollo de este proyecto y la creación del entorno de microservicios, lo primero es desplegar una máquina virtual con el SO Operativo Ubuntu 23.10 utilizando VirtualBox para alojar el entorno de los microservicios.

Las características del servidor Ubuntu son las siguientes:

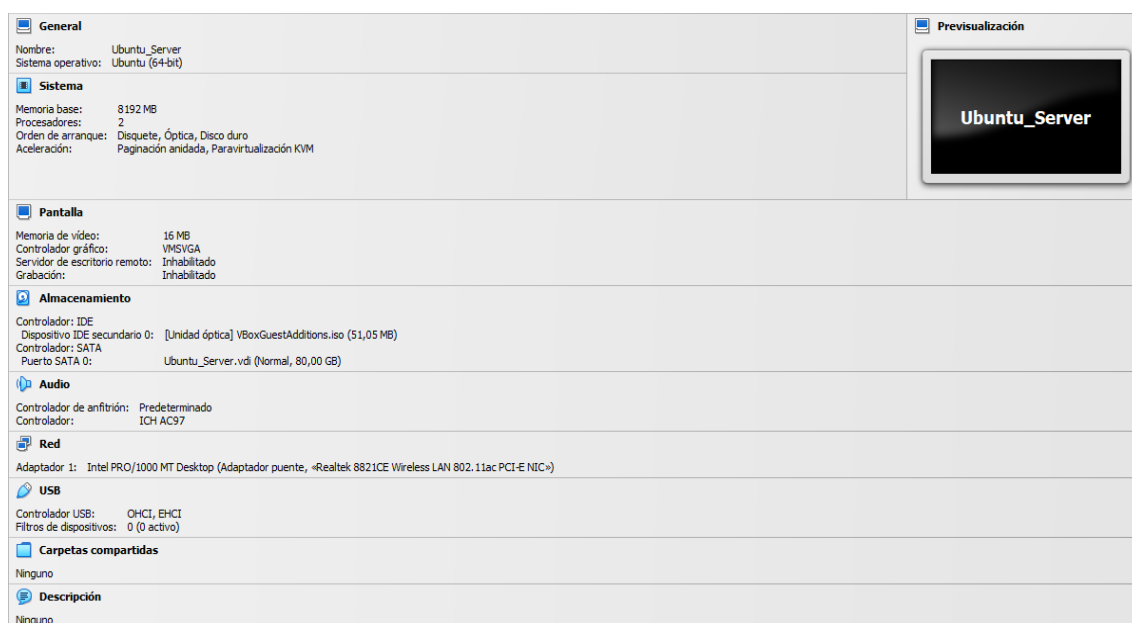


Figura 3 MV Ubuntu (Fuente: propia)

### 5.2. Instalación de Docker

Es necesario instalar Docker en la máquina virtual Ubuntu para la gestión de contenedores. Docker es un proyecto de código abierto que permite automatizar el despliegue de aplicaciones dentro de contenedores, y que, por tanto, nos proporcionará la infraestructura necesaria para ejecutar los microservicios de la aplicación. Aunque no es el único entorno de contenedores, en la actualidad es el más utilizado.

Un contenedor empaqueta de forma ligera todo lo necesario para que uno o más procesos funcionen: código, herramientas y bibliotecas del sistema, dependencias, etc. Esto garantiza que dicho proceso se podrá ejecutar independientemente del entorno en el que se quiera desplegar.

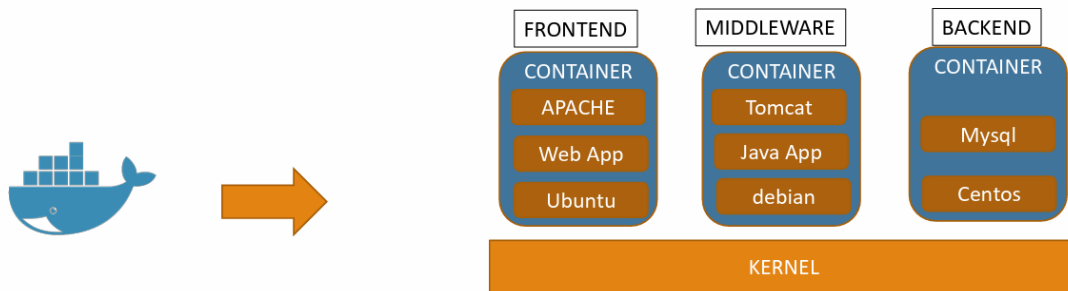


Figura 4 Arquitectura Docker (Fuente: Curso Docker Apasoft)

Dentro de la arquitectura de Docker se destacan los siguientes componentes y elementos a instalar.

- **Docker Engine(docker-ce):** Motor principal de Docker que permite trabajar con imágenes y contenedores.
- **Docker Client (docker-ce-cli):** Es la consola de comandos donde se interactúa con el Docker Engine y se ejecutan los comandos para crear y eliminar imágenes, contenedores, redes, etc.
- **Docker buildx (docker-buildx-plugin):** Plugin que se encarga de la compilación de imágenes.
- **Docker Compose (docker-compose-plugin):** Ya se incorpora como un plugin de Docker, anteriormente se instalaba como un programa aparte. Herramienta para orquestación de aplicaciones multi-contenedor que nos permite definir y gestionar la relación y configuración entre los distintos servicios de la aplicación.
- **Docker Daemon (docker-containerd.io):** Este es un proceso importante para el funcionamiento interno de Docker. Se encarga del almacenamiento de los contenedores, imágenes, logs y redes.
- **Imágenes: plantilla para construir un contenedor, a partir de ellas se construyen los contenedores con los microservicios deseados.**
- **Imagen Registry:** Aunque las imágenes se descargan en local, se traen de un repositorio denominado registro de imágenes. El repositorio de imágenes más utilizado es Docker Hub.
- **Contenedores:** Unidad de software que empaqueta el código y todas las librerías necesarias para poder ejecutar una aplicación o servicio.

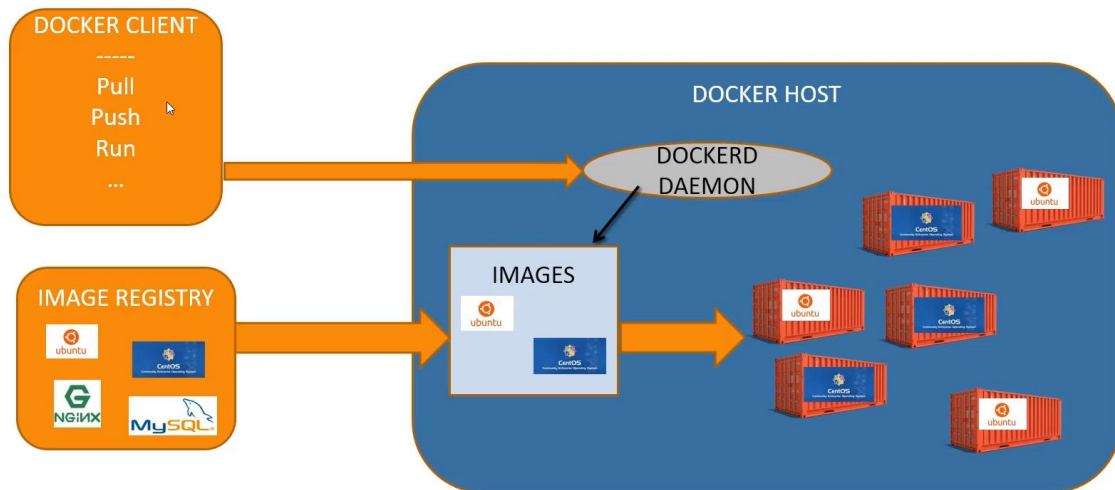


Figura 5 Componentes Docker (Fuente: Curso Docker Apasoft)

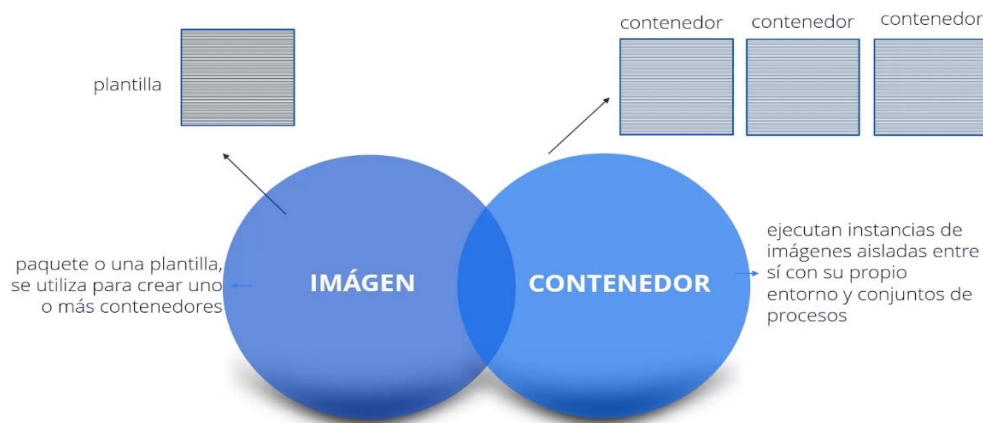


Figura 6 Imágenes vs Contenedores (Fuente: Curso DevOps total)

A continuación, se procede a instalar Docker en la máquina virtual Ubuntu siguiendo los pasos indicados en la página oficial de Docker.

En primer lugar, se desinstalan posibles paquetes antiguos que se pudieran tener de anteriores instalaciones.

```
root@ubuntu:~# for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

Se realiza una actualización de los paquetes del SO.

```

root@ubuntu:~# apt-get update
Obj:1 http://security.ubuntu.com/ubuntu mantic-security InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu mantic InRelease
Des:3 https://dl.google.com/linux/chrome/deb stable InRelease [1.825 B]
Obj:4 http://es.archive.ubuntu.com/ubuntu mantic-updates InRelease
Obj:5 http://es.archive.ubuntu.com/ubuntu mantic-backports InRelease
Des:6 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1.079 B]
Descargados 2.904 B en 1s (2.143 B/s)
Leyendo lista de paquetes... Hecho

```

Se configura el repositorio correcto para Docker instalando algunos paquetes adicionales necesarios para el funcionamiento de Docker.

```

root@ubuntu:~# apt-get install ca-certificates curl
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20230311ubuntu1).
fijado ca-certificates como instalado manualmente.
Se instalarán los siguientes paquetes NUEVOS:
  curl
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 216 kB de archivos.
Se utilizarán 490 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://es.archive.ubuntu.com/ubuntu mantic-updates/main amd64 curl amd64 8.2.1-1ubuntu3.3 [216 kB]
Descargados 216 kB en 1s (264 kB/s)
Seleccionando el paquete curl previamente no seleccionado.
(Leyendo la base de datos ... 147114 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../curl_8.2.1-1ubuntu3.3_amd64.deb ...
Desempaquetando curl (8.2.1-1ubuntu3.3) ...
Configurando curl (8.2.1-1ubuntu3.3) ...
Procesando disparadores para man-db (2.11.2-3) ...

```

Se agregan las claves GPG oficiales de Docker. Estas claves son las que aseguran que cuando se está descargando un paquete de un repositorio, este repositorio es de plena confianza.

```

root@ubuntu:~# install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

```

Se añaden los repositorios de Docker a los repositorios del servidor Ubuntu.

```

root@ubuntu:~# echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
tee /etc/apt/sources.list.d/docker.list > /dev/null

```

Se vuelve a realizar una actualización de los paquetes de SO por si es necesario que tenga que recopilar algo del repositorio que se acaba de instalar.

```

root@ubuntu:~# apt-get update
Obj:1 http://es.archive.ubuntu.com/ubuntu mantic InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu mantic-updates InRelease
Obj:3 http://security.ubuntu.com/ubuntu mantic-security InRelease
Des:4 https://download.docker.com/linux/ubuntu mantic InRelease [48,8 kB]
Obj:5 http://es.archive.ubuntu.com/ubuntu mantic-backports InRelease
Obj:6 https://dl.google.com/linux/chrome/deb stable InRelease
Des:7 https://download.docker.com/linux/ubuntu mantic/stable amd64 Packages [10,1 kB]
Descargados 59,0 kB en 1s (47,0 kB/s)
Leyendo lista de paquetes... Hecho

```

Finalmente, se instala la última versión de Docker Engine ejecutando el siguiente comando.

```

root@ubuntu:~# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  docker-ce-rootless-extras git git-man liberror-perl libslirp0 pigz slirp4netns
Paquetes sugeridos:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
Se instalarán los siguientes paquetes NUEVOS:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git git-man liberror-perl libslirp0 pigz slirp4netns
0 actualizados, 12 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 124 MB de archivos.
Se utilizarán 452 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S

```

Se verifica que Docker está funcionando correctamente.

```

root@ubuntu:~# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-03-29 19:26:38 CET; 3min 19s ago
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 13065 (dockerd)
      Tasks: 9
     Memory: 28.4M
        CPU: 382ms
    CGroup: /system.slice/docker.service
            └─13065 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

```

Se comprueba la versión que se ha instalado tanto de Docker Engine como del orquestador ligero de contenedores Docker Compose.

```

root@ubuntu:~# docker --version
Docker version 26.0.0, build 2ae903e
root@ubuntu:~# docker compose version
Docker Compose version v2.25.0

```

Se le dice al SO que cada vez que arranque inicie automáticamente el servicio de Docker.

```

root@ubuntu:~# systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker

```

## 5.3. Instalación de Ansible

Se debe instalar y configurar Ansible en la máquina virtual para automatizar tareas de mantenimiento y gestión del entorno.

Ansible es una aplicación open source que permite la gestión de infraestructura TI automatizando las tareas más comunes en cualquier sistema informático, como el despliegue de aplicaciones, la gestión y configuración, el aprovisionamiento y la automatización de la infraestructura.

Tiene gran fiabilidad, consistencia y escalabilidad. Utiliza el concepto de *Infrastructure as Code* (IaC) lo que hace muy fácil su uso, ya que permite crear ficheros de configuración llamados Playbooks para poder provisionar y gestionar la infraestructura.

Otra de las características más interesantes de Ansible es que no necesita agentes para trabajar con ellos, solo instalar SSH y el intérprete de comandos Python.



Los componentes principales de Ansible son los siguientes:

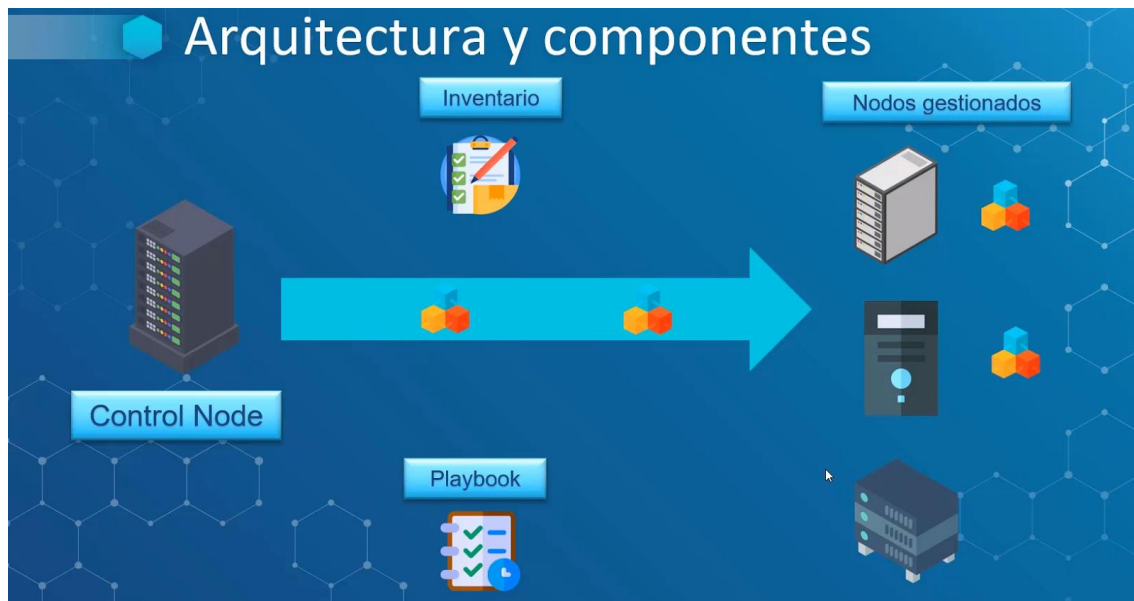


Figura 7 Componentes de Ansible(Fuente: Curso Ansible Apasoft)

- **Control Nodes:** Servidores desde los que se ejecuta Ansible y se lanzan los comandos contra los nodos gestionados. Funcionan sobre entornos Linux. Para Windows se utiliza el Módulo WSL.
- **Manage Nodes:** Servidores que son gestionados por Ansible. Estos nodos están incluidos dentro de un inventario.
- **Inventario:** Es donde se guarda la lista de nodos gestionados y características de estos. Pueden ser estáticos o generarlos a través de scripts.
- **Playbooks y Plays:** Un Play es la serie de tareas que se quieren ejecutar en los nodos gestionados. Se crean en formato YAML y son de tipo declarativo, por lo que son de fácil uso y manejo. Estos Plays se pueden agrupar formando Playbooks.
- **Módulos:** Es una librería con scripts o comandos que se pueden ejecutar dentro de un Playbook. Habrá módulos para gestionar la red, otro para trabajar con ficheros, etc.
- **Colecciones:** Son un formato de distribución de Ansible que está compuesto por múltiples componentes de Ansible, como una distribución de este. En Ansible Galaxy se encuentran módulos y colecciones.

Para instalar Ansible se necesitan los siguientes requisitos:

- Servidor Linux.
- Instalar Python 3.8 o superior.
- Tener instalado SSH dentro del servidor Linux.

Se procede a instalar Ansible siguiendo los pasos indicados en la página oficial. Se va a hacer como root para que la aplicación pueda ser utilizada por varios usuarios.

Primero, se realiza una actualización de los paquetes del SO.

```
root@ubuntuserv:~# apt-get update
```

Se verifica que Python está instalado en una versión correcta.

```
root@ubuntuserv:~# python3 --version
```

Se instala el componente *software-properties-common* que contiene librerías y componentes comunes para el software a instalar en entornos Ubuntu. La mayoría de las veces este software ya viene preinstalado.

```
root@ubuntuserv:~# apt-get install software-properties-common
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
software-properties-common ya está en su versión más reciente (0.99.39).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
```

Se añade el repositorio indicando que busque dentro de la paquetería de Ubuntu un programa llamado Ansible.

```
root@ubuntuserv:~# add-apt-repository --yes --update ppa:ansible/ansible
```

Se procede a instalar Ansible con el siguiente comando.

```
root@ubuntuserv:~# apt install ansible
```

Se verifica la instalación.

```
root@ubuntuserv:~# ansible --version
ansible [core 2.16.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.11.6 (main, Oct 8 2023, 05:06:43) [GCC 13.2.0] (/usr/bin/python3)
  Jinja version = 3.1.2
  libyaml = True
```

## 5.4.Instalación Visual Studio Code

Se instalará Visual Studio Code como el entorno de desarrollo integrado (IDE) para facilitar la escritura y edición de scripts de Ansible y otros archivos de configuración.

El software Visual Studio Code, que es un potente editor para trabajar de manera más sencilla con los distintos ficheros de configuración que normalmente estarán en formato YAML dentro de entornos Docker y Ansible, entre otros. Proporcionará herramientas de edición y depuración que mejorarán la productividad del desarrollo y la gestión del proyecto.

Para su instalación se accede a la página oficial del programa y se descarga la versión correcta para el servidor Ubuntu.

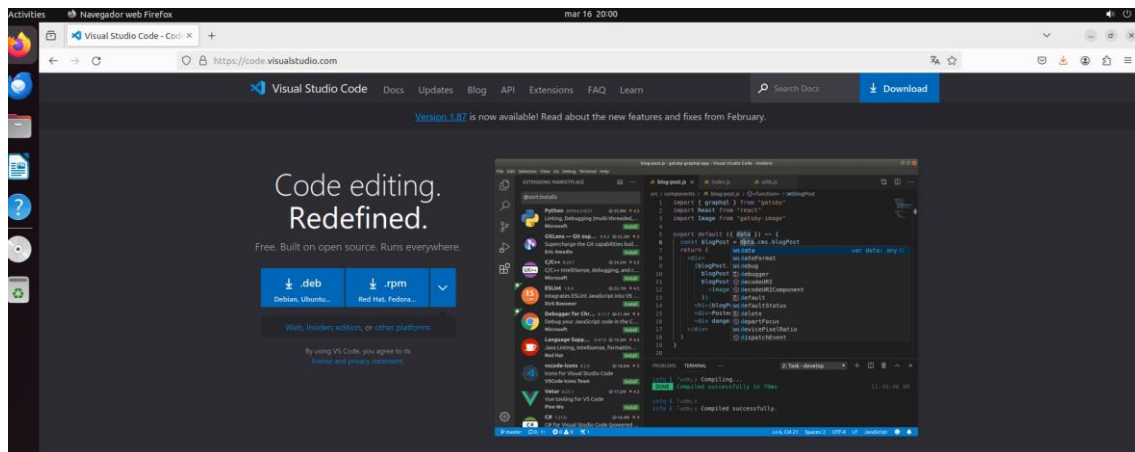


Figura 8 Visual Studio Code (Fuente: propia)

Se verifica que se ha descargado el paquete correcto y se procede a su instalación.

```
root@ubuntu:~/home/molmo/Descargas# ls -la
total 97600
drwxr-xr-x  2 molmo molmo    4096 mar 30 14:07 .
drwxr-xr-x 17 molmo molmo    4096 mar 30 14:03 ..
-rw-rw-r--  1 molmo molmo 99932898 mar 30 14:07 code_1.87.2-1709912201_amd64.deb
root@ubuntu:~/home/molmo/Descargas# dpkg -i code_1.87.2-1709912201_amd64.deb
Seleccionando el paquete code previamente no seleccionado.
(Leyendo la base de datos ... 173988 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar code_1.87.2-1709912201_amd64.deb ...
Desempaquetando code (1.87.2-1709912201) ...
Configurando code (1.87.2-1709912201) ...
Procesando disparadores para mailcap (3.70+nmubuntu1) ...
Procesando disparadores para gnome-menus (3.36.0-1ubuntu1) ...
Procesando disparadores para desktop-file-utils (0.26-1ubuntu5) ...
Procesando disparadores para shared-mime-info (2.2-1) ...
```

Una vez instalado, se abre Visual Studio Code y se instalan las extensiones para trabajar con Docker y Ansible.

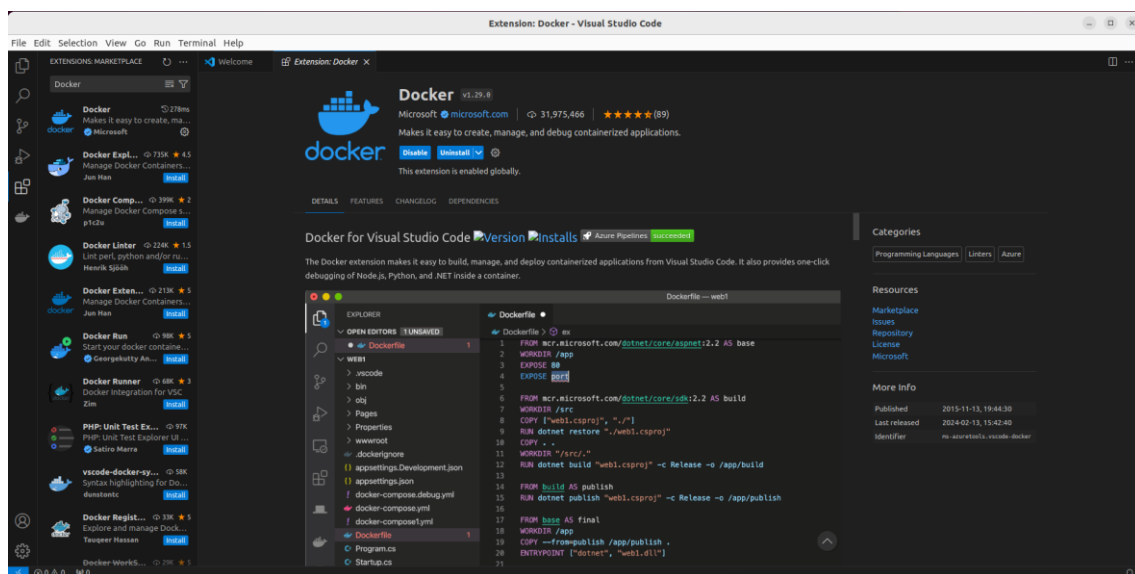


Figura 9 Visual Studio Code extensión Docker (Fuente: propia)

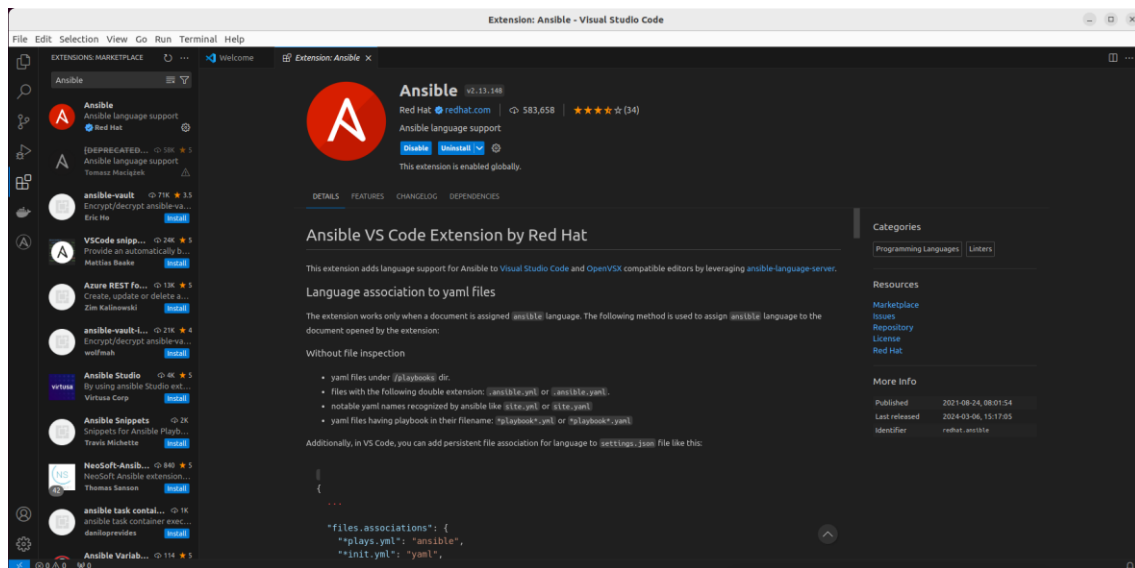


Figura 10 Visual Studio Code extensión Ansible (Fuente :propia)

## 6. Operaciones. Orquestación Docker Compose

En este capítulo se va a generar la orquestación de los contenedores mediante de Docker Compose y la definición de las imágenes a partir de las imágenes oficiales descargadas de Docker Hub y modificadas con las instrucciones de los ficheros Dockerfile necesarias para adaptarlas al entorno.

Una vez definido el entorno, se ejecutará la orquestación verificando que los servicios están levantados. También se subirán las imágenes al repositorio público Docker Hub y se eliminara y recreara la orquestación levantado de nuevo los servicios para mostrar la facilidad con la que se despliegan utilizando herramientas de DevOps.

Para ello, se crea el fichero docker-compose.yml. Este fichero contendrá las instrucciones para generar el entorno multi contenedor. En él se definen los tres servicios/contenedores a desplegar, especificando cómo es cada contenedor y las relaciones que tienen entre ellos, incluyendo las redes, IP, volúmenes y demás propiedades. Se escribe en formato YAML, que es un formato de serialización de datos. Un fichero YAML está compuesto por una serie de pares clave:valor que se incluyen dentro de propiedades y subpropiedades que definirán las características de los servicios a desplegar.

En esta figura se muestra el entorno a desplegar.

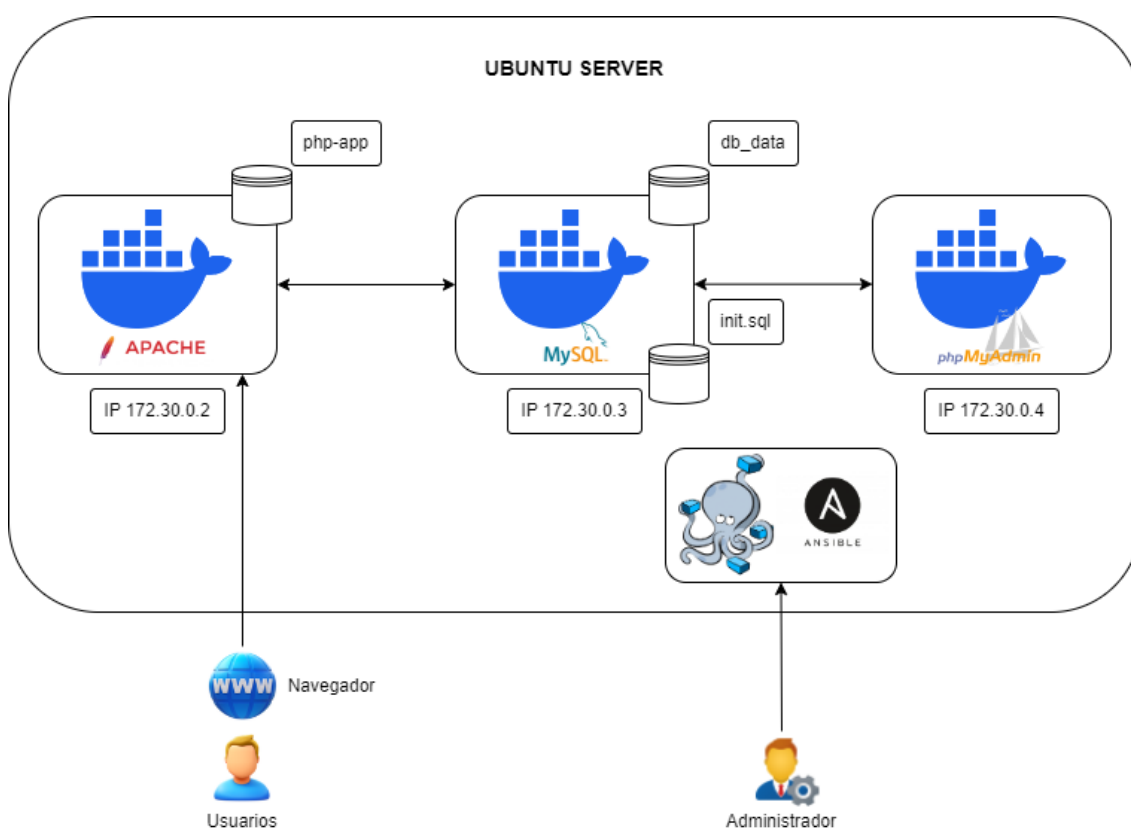


Figura 11 Entorno contenedores y servicios (Fuente: propia)

En los siguientes puntos se describe tanto la creación de la red como los servicios definidos en el fichero de orquestación docker-compose.yml.

## 6.1. Definición de la red Docker

Se define la red para los contenedores.

```
docker-compose.yml > {} services > {} svc-php-apache > {} networks > {} redproy > [x] ipv4_address
docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-ag
1  networks:
2      redproy:
3          driver: bridge
4          ipam:
5              driver: default
6              config:
7                  - subnet: 172.30.0.0/24
```

- Propiedad **“Networks”**: Crea una red tipo bridge para los contenedores con el direccionamiento 172.30.0.0/24.
  - **“driver”**: Define la red de tipo bridge. Este tipo es el controlador de red es el predeterminado en redes Docker. Se utiliza cuando varios contenedores deben comunicarse para ejecutar la aplicación dentro del mismo host.
  - **“ipam”**: (*IP Address Management*): Define la administración de las direcciones IPs de la red en el archivo de configuración docker-compose.yml.

## 6.2. Definición de los servicios

Se definen los servicios de frontend, backend y entorno gráfico phpMyAdmin.

```
9  services:
10     svc-mysql:
11         container_name: mysql
12         image: mysql:latest
13         restart: always
14         environment:
15             MYSQL_ROOT_PASSWORD: mmb1932&
16             MYSQL_DATABASE: facturas
17             MYSQL_USER: adminmysql
18             MYSQL_PASSWORD: mmb1932&
19         volumes:
20             - ./bbdd/db_data:/var/lib/mysql
21             - /proyecto/bbdd/init.sql:/init.sql:/docker-entrypoint-initdb.d/init.sql
22         ports:
23             - "3306:3306"
24         networks:
25             redproy:
26                 ipv4_address: 172.30.0.3
```

```

28  svc-phpmyadmin:
29      container_name: phpmyadmin
30      build:
31          context: ./phpmyadmin
32      restart: always
33      environment:
34          PMA_HOST: svc-mysql
35          PMA_PORT: 3306
36          MYSQL_ROOT_PASSWORD: mmb1932&
37      depends_on:
38          - svc-mysql
39      ports:
40          - "8080:80"
41          - "2222:22"
42      networks:
43          redproy:
44              ipv4_address: 172.30.0.4
45      volumes:
46          - /root/.ssh/id_rsa:/root/.ssh/id_rsa
47          - /root/.ssh/id_rsa.pub:/root/.ssh/id_rsa.pub
48          - /root/.ssh/authorized_keys:/root/.ssh/authorized_keys
49
50  svc-php-apache:
51      container_name: apache
52      build:
53          context: ./php-apache
54      restart: always
55      ports:
56          - "8000:80"
57          - "2223:22"
58      networks:
59          redproy:
60              ipv4_address: 172.30.0.2
61      volumes:
62          - ./php-app:/var/www/html
63          - /root/.ssh/id_rsa:/root/.ssh/id_rsa
64          - /root/.ssh/id_rsa.pub:/root/.ssh/id_rsa.pub
65          - /root/.ssh/authorized_keys:/root/.ssh/authorized_keys

```

A continuación, se explican las instrucciones donde se definen los servicios.

- Propiedad **“services”**: Define el nombre de los servicios en este caso `svc-mysql`, `svc-phpmyadmin` y `svc-php-apache`, con las siguientes propiedades:
  - **“container\_name”**: Indica los nombres de los contenedores a desplegar.
  - **“image”**: Indica la imagen de Docker Hub a utilizar para la construcción del contenedor. Se aplica solo en el caso del contenedor `mysql`, en los contenedores `apache` y `phpMyAdmin` se utilizará un fichero `Dockerfile`.
  - **“build”**: Indica que la imagen del contenedor se construirá a través de un fichero `Dockerfile`. Mediante la propiedad `context` se facilita la ruta del fichero `Dockerfile`.
  - **“restart”**: Define cómo se comportará el contenedor en caso de producirse algún error. Al indicarle el valor `always` se le ordena que se reinicie ante cualquier problema.
  - **“environment”**: Se le pasan las variables de entorno a los servicios en el caso del `svc-mysql` y `svc-phpmyadmin`. En el servicio `svc-mysql` se le pasa el *password*

del usuario root (MYSQL\_ROOT\_PASSWORD), esta variable es obligatoria para la conexión al motor de la BBDD. Se crea una BBDD llamada facturas (MYSQL\_DATABASE) y un usuario de la BBDD con su correspondiente *password* (MYSQL\_USER, MYSQL\_PASSWORD). En el servicio svc-phpmyadmin además del *password* del usuario root (MYSQL\_ROOT\_PASSWORD) obligatorio, las variables PMA\_HOST y PMA\_PORT se usan para especificar el servicio de BBDD y el puerto al que se conectará phpMyAdmin.

- **“ports”**: Indica el puerto externo del host anfitrión, en este caso el servidor Ubuntu y el puerto interno del contenedor. Para que los servicios dockerizados puedan ser accesibles desde fuera deben escuchar por algún puerto externo que se mapeará al puerto interno del contenedor que alberga el servicio. Se añade también el puerto 22 SSH para futuras tareas de automatización con Ansible.
- **“networks”**: Indica al contenedor a que red debe unirse y se asigna dirección IP fija con la propiedad “ipv4\_address”.
- **“volumes”**: Genera los volúmenes para los servicios, con el objetivo de obtener persistencia de los datos. Se generan los siguientes volúmenes.
  - `./bbdd/db_data:/var/lib/mysql` -> volumen para persistencia de datos del servicio MySQL. El directorio local `./bbdd/db_data` (en el servidor Ubuntu donde se está ejecutando Docker) se montará dentro del contenedor en el directorio `/var/lib/mysql`. Este es el directorio predeterminado donde MySQL almacena sus datos y bases de datos.
  - `/proyecto/bbdd/init.sql/init.sql:/docker-entrypoint-initdb.d/init.sql` -> volumen con un script de inicio que generará una tabla llamada facturas con los campos indicados en el script.

```
bbdd > init.sql > init.sql > ...
  Run | New Tab | Active Connection
1  USE facturas;
2
  Run | New Tab | Copy
3  CREATE TABLE IF NOT EXISTS facturas (
4      id INT AUTO_INCREMENT PRIMARY KEY,
5      cliente VARCHAR(255),
6      direccion VARCHAR(255),
7      correo VARCHAR(255),
8      producto VARCHAR(255),
9      cantidad INT,
10     precio DECIMAL(10,2)
11 );
```

- `./php-app:/var/www/html` -> volumen para persistencia de los ficheros de la aplicación de facturación, el directorio local `./php-app` (en el servidor Ubuntu donde se está ejecutando Docker) se montará dentro del contenedor en el directorio `/var/www/html` con los ficheros `index.html` y `guardar_facturas.php`. Este es el directorio predeterminado en un servidor apache donde se almacena el contenido web que se ofrece a los usuarios.



- `/root/.ssh/id_rsa:/root/.ssh/id_rsa,`  
`/root/.ssh/id_rsa.pub:/root/.ssh/id_rsa.pub` y  
`/root/.ssh/authorized_keys:/root/.ssh/authorized_keys->` volúmenes para copiar las claves SSH generadas en el servidor en los contenedores, de manera que se tenga acceso por SSH para las tareas Ansible.

### 6.2.1. Ficheros Dockerfile

Como se indica en el apartado anterior, con la propiedad *build* se le dice al orquestador que la imagen de los contenedores phpMyAdmin y apache se construirá a partir de un fichero Dockerfile. A través de estos ficheros Dockerfile modificamos la imagen oficial descargada de Docker Hub, instalando los paquetes de software necesarios para preparar la gestión y automatización mediante Ansible de los contenedores que se ejecutaran a partir de estas imágenes que se están definiendo.

```
php-apache > Dockerfile > ...
1  # Utiliza la imagen base de PHP con Apache
2  FROM php:apache
3
4  # Update
5  RUN apt-get update
6  # Instala servidor openssh
7  RUN apt-get install -y openssh-server
8  # Instala interprete python
9  RUN apt-get install -y python3
10 # Instala la extensión MySQLi
11 RUN docker-php-ext-install mysqli
12 # Inicia el servicio SSH al arrancar el contenedor
13 CMD service ssh start && apache2-foreground
```

```
phpmyadmin > Dockerfile > ...
1  # Utiliza la imagen base de phpmyadmin
2  FROM phpmyadmin:latest
3
4  # Update
5  RUN apt-get update
6  # Instala servidor openssh
7  RUN apt-get install -y openssh-server
8  # Instala interprete python
9  RUN apt-get install -y python3
10 # Copia el archivo config.secret.inc.php en el contenedor
11 COPY config.secret.inc.php /etc/phpmyadmin/config.secret.inc.php
12 # Inicia el servicio SSH al arrancar el contenedor
13 CMD service ssh start && apache2-foreground
```

- Comando **FROM**: Indica la imagen base a utilizar para generar nuestra propia imagen. En este caso se utiliza la imagen oficial phpapache y phpMyAdmin, ambas se obtienen del repositorio público Docker Hub. Con la etiqueta *latest* se indica que debe utilizar la imagen más reciente del repositorio.
- Comando **RUN**: Con RUN se ejecutan comandos en el proceso de construcción de la imagen Docker. En este caso se lanzan los siguientes comandos.
  - **apt-get update**: Se realiza una actualización de paquetes del SO.
  - **apt-get install -y openssh-server**: Se instala el servidor SSH en el contenedor apache. Ansible se comunicará con los contenedores gestionados a través de SSH, por lo que se instala SSH en el contenedor de apache y phpMyAdmin.
  - **apt-get install -y python3**: Se instala Python en el contenedor. Es necesario tener instalado el intérprete de Python para las tareas esenciales que realiza Ansible, como ejecución de módulos que realizan tareas específicas, recopilación de datos de los contenedores gestionados.
  - **docker-php-ext-install mysqli**: Se instala la extensión mysqli necesaria para la conexión de PHP con la base de datos MySQL.
  - **COPY config.secret.inc.php /etc/phpmyadmin/config.secret.inc.php**: Copiamos el archivo de configuración config.secret.inc.php del servidor Ubuntu al contenedor. Este es un archivo de configuración de phpMyAdmin que no se genera con la imagen phpMyAdmin de Docker Hub en la que se basa el contenedor, por lo tanto, es necesario definirlo y copiarlo al construir nuestra imagen personalizada.
  - **CMD service ssh start && apache2-foreground**: Con el comando CMD se especifican los comandos predeterminados que se ejecutará cuando se inicie un contenedor basado en la imagen del Dockerfile. En este caso primero se inicia el servidor SSH necesario para la conexión de Ansible y después inicia el servidor web Apache en primer plano. Se inicia el proceso principal en primer plano ya que si no el contenedor queda reiniciando continuamente al no encontrar Docker un proceso principal estable. Docker espera un proceso principal (el que está especificado en el CMD apache2-foreground) si no lo encuentra o el proceso principal finaliza, Docker asume que el contenedor ha completado su tarea y lo detiene. Es por esto por lo que en un principio al ejecutar el comando *CMD service ssh start* sin añadir el proceso principal el contenedor se reinicia continuamente.

### 6.3. Ejecución de la orquestación Docker Compose

En este apartado se va a ejecutar la orquestación mediante Docker Compose con el siguiente comando.

```

root@ubuntuserv:/proyecto# docker compose -f docker-compose.yml up -d
[+] Running 11/11
 ✓ svc-mysql 10 layers [██████████] 0B/0B Pulled
 ✓ 2ba873cb070a Pull complete
 ✓ dd1a4da808dd Pull complete
 ✓ 3292fb4adf41 Pull complete
 ✓ 3811c45068cc Pull complete
 ✓ e13320244c05 Pull complete
 ✓ 6a34d702f281 Pull complete
 ✓ de90f4481477 Pull complete
 ✓ d575200ae375 Pull complete
 ✓ aea400be5707 Pull complete
 ✓ 38c930606a4f Pull complete
[+] Building 81.6s (20/20) FINISHED

```

- -f: fichero de la orquestación.
- -d: Ejecutar los contenedores en background.
- Opción up: Que levante todos los servicios y cree la red.

En los logs de la instalación se ve que los servicios svc-php-apache y svc-phpmyadmin vienen definidos a través de un Dockerfile en la orquestación.

```

[+] Building 81.6s (20/20) FINISHED
=> [svc-php-apache internal] load build definition from Dockerfile
=> => transferring dockerfile: 319B
=> [svc-phpmyadmin internal] load build definition from Dockerfile
=> => transferring dockerfile: 322B

```

La descarga de la imagen base para los contenedores.

```

=> [svc-phpmyadmin 1/5] FROM docker.io/phpmyadmin/phpmyadmin:latest@sha256:67ba2558fd004399ab0b95b64021a88ea544011e566a9a1995180a3dec6410d
=> => resolve docker.io/phpmyadmin/phpmyadmin:latest@sha256:67ba2558fd004399ab0b95b64021a88ea544011e566a9a1995180a3dec6410d

```

```

=> [svc-php-apache 1/5] FROM docker.io/library/php:apache@sha256:94e89d509757e8c8566a4bafc8fcca52ec29d9591c8168d0f10bf9e2d2893
=> => resolve docker.io/library/php:apache@sha256:94e89d509757e8c8566a4bafc8fcca52ec29d9591c8168d0f10bf9e2d2893

```

La ejecución de los comandos definidos con RUN en los Dockerfile.

```

=> [svc-phpmyadmin 2/5] RUN apt-get update
=> [svc-phpmyadmin 3/5] RUN apt-get install -y openssh-server
=> [svc-php-apache 2/5] RUN apt-get update
=> [svc-php-apache 3/5] RUN apt-get install -y openssh-server
=> [svc-php-apache 4/5] RUN apt-get install -y python3
=> [svc-phpmyadmin 4/5] RUN apt-get install -y python3
=> [svc-phpmyadmin 5/5] RUN docker-php-ext-install mysqli
=> [svc-php-apache 5/5] RUN docker-php-ext-install mysqli

```

Y el resultado de la orquestación.

```

[+] Running 3/4
 ⋮ Network proyecto_redproy Created
 ✓ Container mysql Started
 ✓ Container apache Started
 ✓ Container phpmyadmin Started

```

Se verifica que se han creado los tres contenedores con los microservicios de Apache, MySQL y Phpmyadmin. Con el comando *docker ps* se ve la información sobre la imagen en la que se basa el contenedor, estado, puerto externo del host, puerto interno del contenedor y nombre que se le ha asignado a cada uno.

```

root@ubuntu:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
81b1f8598262   proyecto-svc-phpmyadmin   "/docker-entrypoint..." 9 minutes ago   Up 9 minutes   0.0.0.0:2222->22/tcp, :::2222->22/tcp, 0.0.0.0:8080->80/tcp, :::8080->80/tcp
7cfa1c9cd312   mysql:latest   "docker-entrypoint.s..." 9 minutes ago   Up 9 minutes   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
2c56f53adfa9   proyecto-svc-php-apache   "docker-php-entrypo..." 9 minutes ago   Up 9 minutes   0.0.0.0:2223->22/tcp, :::2223->22/tcp, 0.0.0.0:8000->80/tcp, :::8000->80/tcp
root@ubuntu:~#

```

Se verifica que el servicio apache está accesible por el puerto 8000 y que se presenta la aplicación web.

The screenshot shows a web browser window with two tabs: 'Formulario de Facturas' and 'phpMyAdmin'. The address bar indicates the URL is 'localhost:8000'. The page content includes a dark blue header with the title 'Formulario de Facturas'. Below the header, there are several input fields for form data: 'Cliente:', 'Dirección:', 'Correo:', 'Producto:', 'Cantidad:', and 'Precio:'. At the bottom of the form, there is a button labeled 'Guardar Factura'.

Figura 12 Frontend (Fuente: propia)

Y el entorno gráfico phpMyAdmin está accesible a través del puerto indicado en fichero docker.compose.yaml localhost:8080.

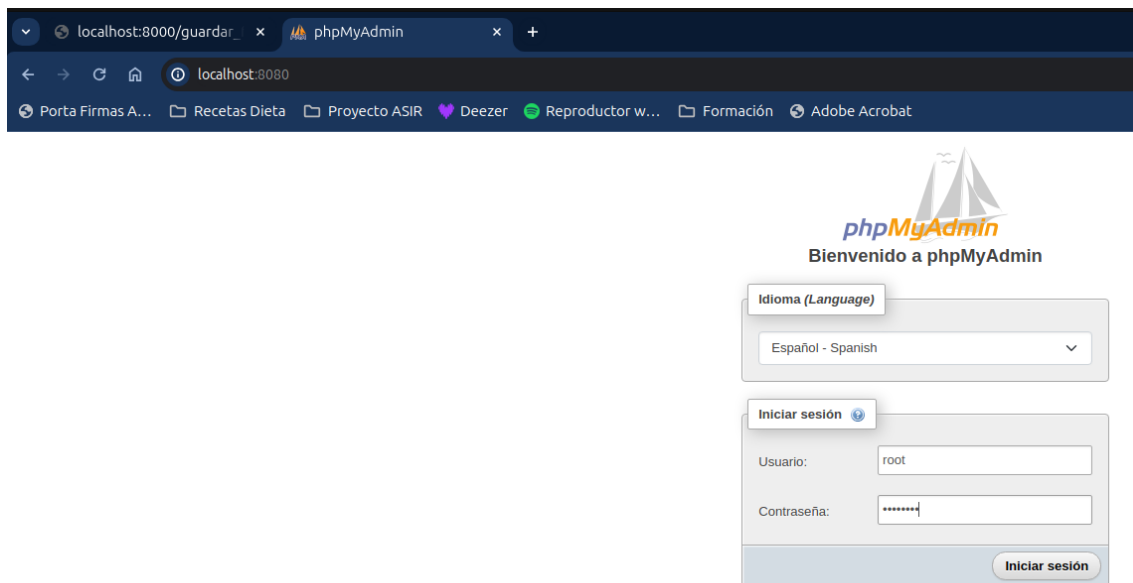


Figura 13 Acceso phpMyAdmin (Fuente: propia)

Se introducen datos de una factura a través del interfaz web frontend.

Figura 14 Datos formulario (Fuente: propia)

Para verificar que los datos introducidos se han guardado, se accede al entorno gráfico phpMyAdmin a través del usuario, *password* y puerto indicado en fichero docker.compose.yaml para comprobar que se ha registrado la factura en la BBDD.

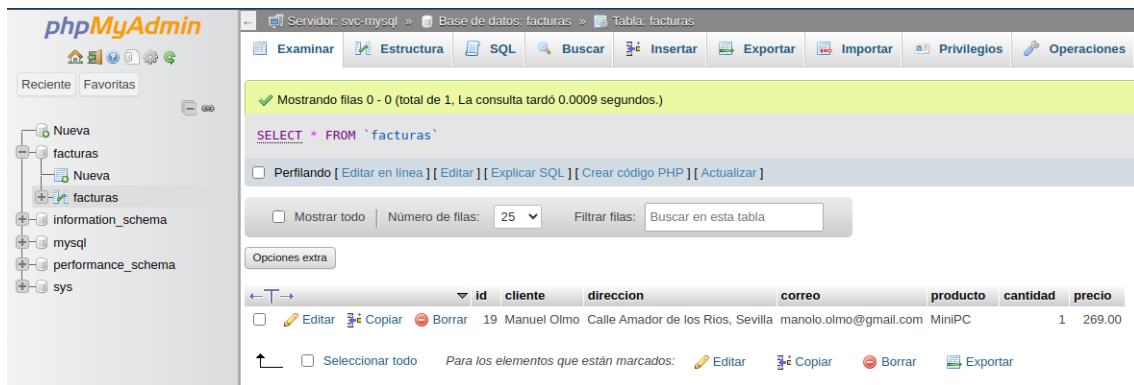


Figura 15 Verificación datos phpMyAdmin (Fuente: propia)

## 6.4.Repositorio de Docker Hub

Se van a subir las imágenes generadas al repositorio público Docker Hub.

En primer lugar, se listan las imágenes que se han creado.

```
root@ubuntuserv:/proyecto# docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
proyecto-svc-phpmyadmin  latest     3dd3117d7b3f  3 hours ago   660MB
proyecto-svc-php-apache  latest     660bed16f76b  3 hours ago   594MB
mysql                latest     65f3f983cb08  2 weeks ago   632MB
```

Para subir las imágenes al repositorio de Docker Hub hay que cumplir la norma del repositorio al nombrar estas imágenes. La norma indica que el nombre de la imagen debe venir precedida del nombre del usuario. En este caso, el usuario Docker Hub es molmo073, por lo tanto, el formato del nombre de las imágenes será molmo073/<nombre\_imagen:etiqueta>.

Con el comando `docker tag` se modifica el nombre de las imágenes para adaptarlas al formato requerido.

```
root@ubuntuserv:/proyecto# docker tag mysql:latest molmo073/mysql:v2
```

```
root@ubuntuserv:/proyecto# docker tag proyecto-svc-php-apache:latest molmo073/apache:v2
```

```
root@ubuntuserv:/proyecto# docker tag proyecto-svc-phpmyadmin:latest molmo073/phpmyadmin:v2
```

Quedando el listado de imágenes como sigue. En este caso se ha utilizado la etiqueta v2 para indicar la versión ya que en anteriores pruebas de creación de imágenes durante la ejecución del proyecto se ha subido una primera versión de estas etiquetadas con v1.

```
root@ubuntuserv:/proyecto# docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
proyecto-svc-phpmyadmin  latest     3dd3117d7b3f  3 hours ago   660MB
molmo073/phpmyadmin    v2         3dd3117d7b3f  3 hours ago   660MB
proyecto-svc-php-apache  latest     660bed16f76b  3 hours ago   594MB
molmo073/phpapache     v2         660bed16f76b  3 hours ago   594MB
molmo073/mysql         v2         65f3f983cb08  2 weeks ago   632MB
mysql                latest     65f3f983cb08  2 weeks ago   632MB
```

A continuación, se tiene que realizar *login* en Docker Hub con el siguiente comando *docker login* al no indicarle el repositorio por defecto intenta logarse en Docker Hub.

```
root@ubuntuserv:/proyecto# docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub
You can log in with your password or a Personal Access Token (PAT). Using a limit
/docs.docker.com/go/access-tokens/

Username: molmo073
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ubuntuserv:/proyecto#
```

Y con el comando *docker push* se suben las imágenes al registro en el que se ha hecho el *login* previamente.

Primero se sube la imagen MySQL.

```
root@ubuntuserv:/proyecto# docker push molmo073/mysql:v2
The push refers to repository [docker.io/molmo073/mysql]
06b1c37f8a6f: Mounted from library/mysql
8d477abc77e5: Mounted from library/mysql
dff1f1ca0c1e: Mounted from library/mysql
19bc5ba2cbe0: Mounted from library/mysql
6127c8b5215b: Mounted from library/mysql
b22c87aab7de: Mounted from library/mysql
20e513fcf277: Mounted from library/mysql
5feabba0cd9b: Mounted from library/mysql
35ed43201134: Mounted from library/mysql
4c023aac3fa2: Mounted from library/mysql
v2: digest: sha256:21b1517c999bcbc53c59b325b7ee8be0b2fb9be1a6b63849fa100054fc48a3d2 size: 2411
```

A continuación, se sube la imagen de phpapache.

```
root@ubuntuserv:/proyecto# docker push molmo073/phpapache:v2
The push refers to repository [docker.io/molmo073/phpapache]
27ba96d470e8: Mounted from molmo073/apache
f6e2890e3162: Mounted from molmo073/apache
8cbf47d9184c: Mounted from molmo073/apache
5a2da7bf1ba0: Mounted from molmo073/apache
ec855582420b: Mounted from molmo073/apache
8f9208273926: Mounted from molmo073/apache
4e089f4244c0: Mounted from molmo073/apache
60b9b666f7d6: Mounted from molmo073/apache
2ac84eb9c511: Mounted from molmo073/apache
98145fa8ad9b: Mounted from molmo073/apache
4b713f2e5bc0: Mounted from molmo073/apache
8eedaabd7bf7: Mounted from molmo073/apache
05769640ee18: Mounted from molmo073/apache
ee50ac85ebd7: Mounted from molmo073/apache
781dffa2223b: Mounted from molmo073/apache
1706a4264cc9: Mounted from molmo073/apache
1f00ff201478: Mounted from molmo073/apache
v2: digest: sha256:6ebc85225f913ebe66a7db83f85606402659126995bd05822ae4bf013b61a2c0 size: 3881
```

Y, por último, se sube la imagen de Phpmyadmin.

```

root@ubuntu:~/proyecto# docker push molmo073/phpmyadmin:v2
The push refers to repository [docker.io/molmo073/phpmyadmin]
e8aae0e87d8b: Pushed
a9dd83bd91d8: Pushed
6b79d949ddf2: Pushed
b93d8990f253: Pushed
c8dd0e06483d: Layer already exists
09ef9e198a0c: Layer already exists
b073bec499d7: Layer already exists
50c77db82438: Layer already exists
0580885321fc: Layer already exists
937b931f3d83: Layer already exists
3c6a0869f4e6: Layer already exists
b33cdd1a9bfc: Layer already exists
c41bbaf795ec: Layer already exists
d25256007733: Layer already exists
3641ab10984b: Layer already exists
c6c385ceab87: Layer already exists
af169763ac2b: Layer already exists
abf6306dfff5: Layer already exists
1dfeaa791054: Layer already exists
bd5b7b7ab346: Layer already exists
90aa21fb5d09: Layer already exists
24839d45ca45: Layer already exists
v2: digest: sha256:d0ff00f2474cc2ced02e62e53c52c615c569009b5990d00128ec508febcd5390 size: 4926

```

Para comprobar que efectivamente las imágenes se han subido correctamente al repositorio de Docker Hub habrá que logarse y se verán las imágenes que han sido subidas.

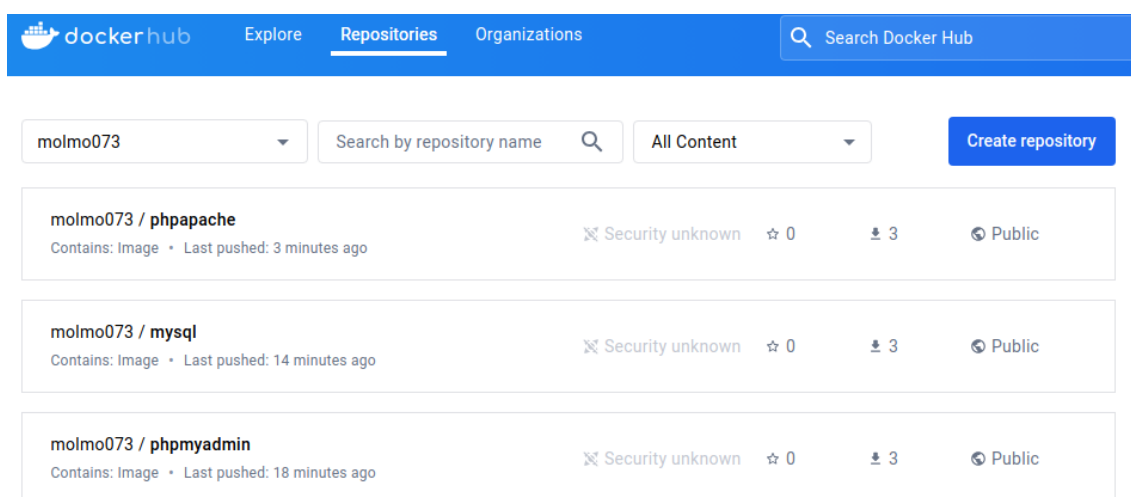


Figura 16 Repositorio Docker Hub (Fuente: propia)

Se entra en cualquiera de ellas y en la pestaña tag se ve la imagen que puede ser descargada para desplegarla en otro servidor. Además, se ven las versiones de la imagen que se han ido subiendo, en este caso, como se comentó anteriormente ya se había subido una primera versión de las imágenes durante el desarrollo del proyecto. Indicando en la etiqueta el número de versión se puede llevar un control de versiones.



**molmo073/phpapache**

Updated 5 minutes ago

Imagen phpapache para proyecto ASIR

**WEB SERVERS**

**Docker commands** [Public View](#)

To push a new tag to this repository:

```
docker push molmo073/phpapache:tagname
```

**Tags**

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
<b>v2</b>		Image	---	5 minutes ago
<b>v1</b>		Image	11 days ago	11 days ago

**Automated Builds**

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).

[Upgrade](#)

Figura 17 Docker Hub versiones apache (Fuente: propia)

## 6.5. Eliminación y recreación de la orquestación

Una de las principales ventajas de implementar metodología DevOps en la entrega de software es que lleva a una mayor eficiencia y entrega más rápida del software como resultado de la automatización de las tareas, esto se conoce como entrega continua. Para demostrar esto se va a eliminar toda la orquestación, servicios e imágenes creadas y se recrearán de nuevo con tan solo un comando, verificando también la persistencia de los datos al utilizar volúmenes.

En primer lugar, se detiene la orquestación del Docker compose.

```
root@ubuntuserv:/proyecto# docker compose stop
[+] Stopping 3/3
✓ Container apache      Stopped
✓ Container phpmyadmin  Stopped
✓ Container mysql       Stopped
```

Se eliminan contenedores, imágenes y volúmenes.

```

root@ubuntuserv:/proyecto# docker system prune --all
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
bcfaaaa006d5a41fed55a04fa3f49e6a897bd1d82e3bd1b05e0bb5afe753a2f5
9564f8cb60b28f9ebba503a8d8dc24b5742e0977f55f8721de02201eb063cad6
956386dfb61c0622e4e47a63b04c2d2dab2bf881d2425fec8c0104a3a2580efb

Deleted Networks:
proyecto_redproy

Deleted Images:
untagged: proyecto-svc-phpmyadmin:latest
untagged: molmo073/phpmyadmin:v2
untagged: molmo073/phpmyadmin@sha256:d0ff00f2474cc2ced02e62e53c52c615c569009b5990d00128ec508febcd5390

```

Se verifica que no queda ninguna imagen ni contenedor.

```

root@ubuntuserv:/proyecto# docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
root@ubuntuserv:/proyecto#

root@ubuntuserv:/proyecto# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@ubuntuserv:/proyecto#

```

Y se vuelve a ejecutar de nuevo toda la orquestación.

```

root@ubuntuserv:/proyecto# docker compose -f docker-compose.yml up -d
[+] Running 11/11
✓ svc-mysql Pulled

[+] Running 1/1
✓ Network proyecto_redproy Created
✓ Container apache Started
✓ Container mysql Started
✓ Container phpmyadmin Started

```

## 6.6. Recreación del entorno en otro sistema

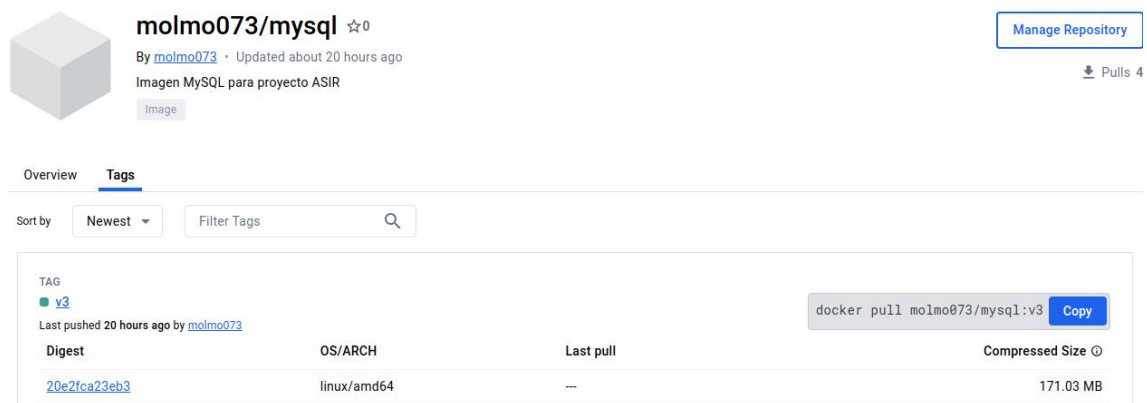
Para demostrar la facilidad de recrear un entorno utilizando tecnologías DevOps en un nuevo sistema TI se ha creado una nueva máquina virtual con el nuevo sistema operativo Ubuntu 24.04 al que le hemos instalado Docker.

```

root@ubuntuserv:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 24.04 LTS
Release: 24.04
Codename: noble

```

Una vez instalado Docker se procede a descargar las imágenes que anteriormente subimos al repositorio Docker Hub. En Docker Hub se puede copiar incluso el comando a utilizar *docker pull*.



**molmo073/mysql** ☆0

By [molmo073](#) · Updated about 20 hours ago

Imagen MySQL para proyecto ASIR

image

Manage Repository

Pulls 4

Overview **Tags**

Sort by **Newest** Filter Tags

TAG	Digest	OS/ARCH	Last pull	Compressed Size
<b>v3</b> Last pushed 20 hours ago by <a href="#">molmo073</a>	<a href="#">20e2fca23eb3</a>	linux/amd64	---	171.03 MB

docker pull molmo073/mysql:v3 [Copy](#)

Figura 18 Docker Hub descarga imagen MySQL (Fuente: propia)

Copiamos el comando para la descarga de las imágenes.

```
root@ubuntuserv:~# docker pull molmo073/mysql:v3
v3: Pulling from molmo073/mysql
bd37f6d99203: Pull complete
aafe9b2df4d0: Pull complete
fc98c884ffd9: Pull complete
587e0e3e0c51: Pull complete
64520c05a6b1: Pull complete
13d3ebffe534: Pull complete
9be2eca8f885: Pull complete
810320f66957: Pull complete
8c425b59d0ec: Pull complete
7387b81cae91: Pull complete
Digest: sha256:20e2fca23eb33d132255d942bfbd99371833f0ad5a00b3301b2eb6e367059bf
Status: Downloaded newer image for molmo073/mysql:v3
```

Descarga de la imagen phpapache.

```
root@ubuntuserv:~# docker pull molmo073/phpapache:v3
v3: Pulling from molmo073/phpapache
b0a0cf830b12: Pull complete
c93478d47932: Pull complete
e74cc574d0d2: Pull complete
e4782e138a90: Pull complete
cfeec87621ae: Pull complete
c1badcd002c0: Pull complete
e0d463a60cb6: Pull complete
c67111b81cd8: Pull complete
e9417e728193: Pull complete
8774e168fe72: Pull complete
5d72544eef59: Pull complete
f0a6ff90fc84: Pull complete
b6c1f8c453b5: Pull complete
a43e9e9f6661: Pull complete
f8e3fc39a3b5: Pull complete
afacbef45922: Pull complete
6c32929b9316: Pull complete
Digest: sha256:c377d3faa8b606990209ecc71416268e57823de757920979e070f265f5ec7e6e
Status: Downloaded newer image for molmo073/phpapache:v3
docker.io/molmo073/phpapache:v3
```

Descarga de la imagen phpMyAdmin.

```
root@ubuntuserv:~# docker pull molmo073/phpmyadmin:v3
v3: Pulling from molmo073/phpmyadmin
b0a0cf830b12: Already exists
c93478d47932: Already exists
e74cc574d0d2: Already exists
e4782e138a90: Already exists
cfeec87621ae: Already exists
c1badcd002c0: Already exists
e0d463a60cb6: Already exists
d1ad50b335f5: Pull complete
98c64971444a: Pull complete
ba7d8962b7e1: Pull complete
e620d8fafd56: Pull complete
2038239aa3e1: Pull complete
df99eaff6870: Pull complete
1d61e18908d9: Pull complete
a503059f17a4: Pull complete
fb51dbc98e8f: Pull complete
a1b6a14258b3: Pull complete
cf8ad639c873: Pull complete
498340219751: Pull complete
377ad19a10c0: Pull complete
d815f433c6ad: Pull complete
d4d7502dbbfe: Pull complete
Digest: sha256:a2c95a40ae474920d45b4ea6f5c535f9ba506da3d7f0513fe66c3df6c352b7e8
Status: Downloaded newer image for molmo073/phpmyadmin:v3
docker.io/molmo073/phpmyadmin:v3
```

Se verifica que se han descargado las tres imágenes desde Docker Hub.

```
root@ubuntu:~# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
molmo073/phpapache	v3	1750dd07c424	21 hours ago	594MB
molmo073/phpmyadmin	v3	fc70700fb82a	21 hours ago	645MB
molmo073/mysql	v3	8251f0669c6e	4 days ago	623MB

Y se lanzan los contenedores con el comando *docker run*.

Apache.

```
root@ubuntu:~# docker run -d --name web_facturas -p 8000:80 molmo073/phpapache:v3
f3928ec65dc15703fb0dc72da47a8054ca490fad7bcc1b22ad97cb4057b2f2d9
```

MySQL.

```
root@ubuntu:~# docker run -d --name MySQL -p 3306:3306 -e MYSQL_ROOT_PASSWORD="mmb19328" molmo073/mysql:v3
aed3d1b5a1d865ad8462d51fd78fa4c3cde4336210a160577aefba2fc9a294e5
```

Y phpMyAdmin.

```
root@ubuntu:~# docker run -d --name phpMyAdmin --link MySQL:db -p 8080:80 molmo073/phpmyadmin:v3
cf777c6ba828628c14de9fdc47eec63082a652f9f94548eac12895ff57986319
```

Se verifica que los contenedores están activos y ejecutándose los servicios.

```
root@ubuntu:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
cf777c6ba828	molmo073/phpmyadmin:v3	"/docker-entrypoint..."	2 minutes ago	Up 2 minutes	0.0.0.0:8080->80/tcp, :::8080->80/tcp	phpMyAdmin
aed3d1b5a1d8	molmo073/mysql:v3	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp	MySQL
f3928ec65dc1	molmo073/phpapache:v3	"docker-php-entrypoi..."	3 minutes ago	Up 3 minutes	0.0.0.0:8000->80/tcp, :::8000->80/tcp	web_facturas

Se copia el index.html en el directorio /var/www/html.

```
root@ubuntu:~# docker cp index.html web_facturas:/var/www/html
Successfully copied 3.07kB to web_facturas:/var/www/html
```

Se verifica que se presentan los servicios apache y phpMyAdmin en el nuevo servidor Ubuntu 24.04.



Figura 19 Verificación servicios en nuevo servidor (Fuente: propia)

Hay que indicar que al recrear el escenario en otro sistema con las imágenes de Docker Hub y lanzar los contenedores con Docker run, hay que asegurarse de que todos los archivos y recursos necesarios para tu aplicación estén disponibles en el nuevo sistema, como ficheros index.html, ficheros .php, script de inicialización de la base de datos. Lo ideal es que estos archivos estén disponibles en un repositorio Git o en un sitio web público, puedes descargarlos utilizando herramientas como *git clone* o *wget*.

## 7. Desarrollo. Frontend HTML, PHP

En este capítulo se describirán los ficheros en los que se ha desarrollado la sencilla aplicación HTML y PHP de facturación. En primer lugar, el fichero index.html donde se estaría el código del frontend o formulario para introducir los datos de la factura y, en segundo lugar, el fichero generar\_facturas.php para la conexión del frontend con la base de datos MySQL y las instrucciones SQL necesarias para dar de alta el registro en la tabla de la BBDD factura.

### 7.1.Frontend. Index.html

A continuación, se describe el fichero index.html.

```
php-app > index.html > html > body > form
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Formulario de Facturas</title>
7  </head>
8  <body>
9      <h2>Formulario de Facturas</h2>
10     <form method="POST" action="guardar_facturas.php">
11         <label for="cliente">Cliente:</label><br>
12         <input type="text" id="cliente" name="cliente"><br>
13
14         <label for="direccion">Dirección:</label><br>
15         <input type="text" id="direccion" name="direccion"><br>
16
17         <label for="correo">Correo:</label><br>
18         <input type="email" id="correo" name="correo"><br>
19
20         <label for="producto">Producto:</label><br>
21         <input type="text" id="producto" name="producto"><br>
22
23         <label for="cantidad">Cantidad:</label><br>
24         <input type="number" id="cantidad" name="cantidad"><br>
25
26         <label for="precio">Precio:</label><br>
27         <input type="number" step="0.01" id="precio" name="precio"><br><br>
28
29         <input type="submit" value="Guardar Factura">
30     </form>
31 </body>
32 </html>
```

Se presenta una aplicación sencilla, ya que el objeto de este proyecto se centra más en la parte de operaciones, que consta de un formulario que utiliza el método POST para el envío de los datos obtenidos al fichero guardar\_factura.php, que es el encargado de tratar los datos obtenidos de la factura.

A continuación, se enumeran y explican con más detalle los atributos del formulario etiqueta `<form></form>`.

- `<form></form>`
  - atributo *action*: Se indica el fichero donde se mandarán los datos para su tratamiento.
  - atributo *method*: Se indica que el método para enviar la información es POST.
- `<label>`: Con la etiqueta `<label>` se define el texto con el que se va a indicar al usuario qué dato se le está solicitando que introduzca.
- `<input>`: Con la etiqueta `<input>` se especifica el campo de entrada donde el usuario puede ingresar los datos.
  - atributo *type*: Se especifica el tipo de control que se mostrará en la interfaz web.
    - *text*: Caja de texto para que el usuario pueda introducir texto (opción por defecto).
    - *email*: Campo de entrada para introducir una dirección de correo electrónico.
    - *number*: Campo de entrada para introducir números.
    - *submit*: Crea un botón para poder enviar los datos del formulario con el texto “Guardar Factura”.
  - atributo *id*: Se identifica de manera única cada campo del formulario dentro del documento html.
  - atributo *name*: Especifica el nombre del control y la pareja clave/valor que se enviará al servidor a través de fichero php que tratará los datos.



## 7.2.Frontend. guardar\_facturas.php

A continuación, se describe el fichero guardar\_facturas.php que recopila la información recogida por el método POST en el formulario HTML en variables para posteriormente guardar estos datos en la BBDD MySQL.

```
php-app > guardar_facturas.php
1  <?php
2  if ($_SERVER["REQUEST_METHOD"] == "POST") {
3      $cliente = $_POST["cliente"];
4      $direccion = $_POST["direccion"];
5      $correo = $_POST["correo"];
6      $producto = $_POST["producto"];
7      $cantidad = $_POST["cantidad"];
8      $precio = $_POST["precio"];
9      if (empty($cliente) || empty($direccion) || empty($correo) || empty($producto) || empty($cantidad) || empty($precio)) {
10         echo "Todos los campos son obligatorios.";
11     } else {
12         $conexion = new mysqli("svc-mysql", "root", "mmb1932&", "facturas");
13         if ($conexion->connect_error) {
14             die("Error al conectar a la base de datos: " . $conexion->connect_error);
15         }
16         $sql = "INSERT INTO facturas (cliente, direccion, correo, producto, cantidad, precio) VALUES (?, ?, ?, ?, ?, ?)";
17         $stmt = $conexion->prepare($sql);
18         $stmt->bind_param("ssssdi", $cliente, $direccion, $correo, $producto, $cantidad, $precio);
19         if ($stmt->execute()) {
20             echo "Factura guardada correctamente.";
21         } else {
22             echo "Error al guardar la factura: " . $conexion->error;
23         }
24         $stmt->close();
25         $conexion->close();
26     }
27 } else {
28     header("Location: index.html");
29     exit();
30 }
31 echo "<br><a href='index.html'>Volver a la página principal</a>";
32 ?>
```

- Con el comando `if ($_SERVER["REQUEST_METHOD"] == "POST")` Se verifica si se han recibido los datos desde el formulario HTML.
- `$cliente = $_POST["cliente"];` Se asigna el valor recibido a una variable PHP llamada cliente. El índice "cliente" debe ser igual que el atributo name del formulario HTML. Se repite la asignación con el resto de los campos del formulario.
- `if (empty($cliente) || empty($direccion) || empty($correo) || empty($producto) || empty($cantidad) || empty($precio)) { echo "Todos los campos son obligatorios."; }` {els... Verifica que se han recibido todos los campos del formulario.
- `$conexion = new mysqli("svc-mysql", "root", "mmb1932&", "facturas");` Se realiza la conexión a la BBDD facturas.
- `if ($conexion->connect_error) { die("Error al conectar a la base de datos: " . $conexion->connect_error); }` Verifica conexión, si hay error se detiene el script y sale el mensaje de error.
- `$sql = "INSERT INTO facturas (cliente, direccion, correo, producto, cantidad, precio) VALUES (?, ?, ?, ?, ?, ?)";` Crea una consulta de inserción SQL que insertará valores en las columnas de la tabla facturas, utilizando marcadores de posición para los valores que serán proporcionados después en el script.
- `$stmt = $conexion->prepare($sql);` Prepara la consulta SQL especificada en \$sql para su ejecución posterior y almacena la declaración preparada en la variable \$stmt.
- `$stmt->bind_param("ssssdi", $cliente, $direccion, $correo, $producto, $cantidad, $precio);` Enlaza valores a los marcadores de posición de una consulta preparada.

`bind_param` es un método para enlazar valores a los marcadores de posición de la consulta preparada. “ssssi”, indica el tipo de valores a enlazar “s”->*string* “d”->*double* “i”->*integer*. Y finalmente las variables con los valores a enlazar a los marcadores de posición.

- **if (\$stmt->execute()) { echo "Factura guardada correctamente.;" } else { echo "Error al guardar la factura: " . \$conexion->error; }** Se ejecuta la consulta y muestra un mensaje dependiendo del resultado.
- **\$stmt->close();** Se cierra la consulta.
- **\$conexion->close();** Se cierra la conexión con la BBDD.
- **header("Location: index.html"); exit();** Si no se reciben datos se redirecciona al index.html.
- **echo '<br><a href="index.html">Volver a la página principal</a>';** Se muestra el enlace para que el usuario vuelva a la página principal index.html.

## 8. Automatización de tareas con Ansible

En este capítulo se realizan las tareas de mantenimiento e instalación a través de Ansible. Para ello se realizan las siguientes tareas.

- Generar claves SSH de los contenedores gestionados.
- Creación de fichero de inventario y archivo de configuración ansible.cfg.
- Creación de tareas Ansible (Playbook).

### 8.1. Generar claves SSH de los contenedores gestionados

Para preparar los contenedores gestionados y que sean accesibles desde Ansible se deben generar un par de claves público/privadas. Estas claves son las que se les pasan a los contenedores al montar los volúmenes para permitir la conexión por SSH.

```
- /root/.ssh/id_rsa:/root/.ssh/id_rsa  
- /root/.ssh/id_rsa.pub:/root/.ssh/id_rsa.pub  
- /root/.ssh/authorized_keys:/root/.ssh/authorized_keys
```

Por lo tanto, es importante generar las claves SSH antes de ejecutar la orquestación con docker compose para que tener ya generadas las claves SSH y se puedan copiar a los contenedores al montar los volúmenes.

```
root@ubuntuserv:/proyecto# ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/root/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /root/.ssh/id_rsa  
Your public key has been saved in /root/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:Ak06fS8iA7zztZVGm1oz2AZAy4+wlgFl+xMV0Eoxjgs root@ubuntuserv  
The key's randomart image is:  
+---[RSA 3072]-----+  
|.o.=+o.|  
|.o*oo.|  
|Eo+=.|  
|..===+|  
|.o.+oo.S|  
|... o=o+|  
| oo .o.%.|  
| oo..B.o|  
|. o|  
+----[SHA256]-----+
```

Se generan dos ficheros .rsa que contiene la clave privada y rsa.pub que contiene la clave pública en un directorio /.ssh que se crea automáticamente si no estaba anteriormente creado, dentro del directorio del usuario con el que se han generado las claves.

Se agrega el contenido de un archivo llamado id\_rsa.pub al final de otro archivo llamado authorized\_keys.

```
root@ubuntuserv:~/.ssh# cat id_rsa.pub >> authorized_keys
```

El archivo **id\_rsa.pub** contiene la clave pública RSA. Esta clave se utiliza para autenticar la conexión SSH.

En el archivo **authorized\_keys** se almacenan las claves públicas autorizadas para autenticar las conexiones SSH. Cada vez que un usuario intenta conectarse al servidor SSH, su clave pública se compara con las claves almacenadas en este archivo para determinar si se permite el acceso.

Como se ha indicado antes estos archivos se montan en los contenedores que necesitan ser accedidos a través de SSH para poder ser gestionados por Ansible.

```
root@ubuntuserv:~/.ssh# ls -lsa
total 28
4 drwx-----  2 root root 4096 may  4 18:23 .
4 drwx----- 10 root root 4096 may  4 14:53 ..
4 -rw-r--r--   1 root root  569 may  1 21:17 authorized_keys
4 -rw-----   1 root root 2602 abr 14 20:44 id_rsa
4 -rw-r--r--   1 root root  569 abr 14 20:44 id_rsa.pub
```

## 8.2.Creación de fichero de inventario y ansible.cfg

A continuación, se empieza a trabajar con ANSIBLE automatizando unas sencillas tareas sobre los contenedores docker apache y phpMyAdmin. Para ello se realizan las siguientes tareas.

- Generar fichero de inventario.
- Modificar fichero de configuración ansible.cfg.

Lo primero que hay que hacer es crear un fichero de inventario, que no es más que un fichero que contiene las máquinas con las que queremos trabajar.

```
ansible > ! inventario.yml > ...
1  all:
2    hosts:
3      apache:
4        ansible_host: 172.30.0.2
5      mysql:
6        ansible_host: 172.30.0.3
7      phpmyadmin:
8        ansible_host: 172.30.0.4
```

A continuación, se crea un fichero de configuración llamado ansible.cfg que es la principal forma que tenemos para decirle a Ansible como tiene que comportarse. En este fichero podemos encontrar los valores por defecto de Ansible y está compuesto por pares de propiedad: valor. El fichero ansible.cfg al haber realizado la instalación a través del gestor de paquetes *apt* se encuentra en el directorio global */etc/ansible* como podemos comprobar con *ansible --version*.

```
root@ubuntuser:~/proyecto/ansible# ansible --version
ansible [core 2.16.5]
  config file = /etc/ansible/ansible.cfg
```

Éste es el contenido del fichero de configuración.

```
root@ubuntuser: /etc/ansible
GNU nano 7.2
[defaults]
inventory=/proyecto/ansible/inventario.yml
interpreter_python = auto_legacy
ansible.cfg
```

- **Inventory:** Le dice a Ansible donde encontrar el fichero de inventario con nombres y las IPs de los contenedores de esta forma no hay que especificar el fichero cada vez que ejecutemos un playbook.
- **interpreter\_python:** con el valor `auto_legacy` se le indica a Ansible que intente automáticamente determinar el intérprete Python más adecuado en los contenedores gestionados.

### 8.3.Creación de tareas en Ansible (Playbook)

Por último, se crea un playbook que verifique que los contenedores están *up* a través de un *ping* y que realice un *update* y *upgrade* de los mismos para mantenerlos actualizados.

```
ansible > ≡ update.yml
1  - name: Ping a máquinas Docker
2    hosts: apache:phpmyadmin
3    tasks:
4      - name: Ejecutar ping a contenedor
5        ansible.builtin.ping:
6
7      - name: Actualizar paquetes en el sistema
8        ansible.builtin.apt:
9          update_cache: yes
10         upgrade: yes
```

Se ejecuta el playbook. Y dará información sobre el resultado de la ejecución de las tareas que ha ejecutado.

```

root@ubuntuserv:/proyecto/ansible# ansible-playbook update.yml

PLAY [Ping a máquinas Docker] *****

TASK [Gathering Facts] *****
ok: [apache]
ok: [phpmyadmin]

TASK [Ejecutar ping a contenedor] *****
ok: [apache]
ok: [phpmyadmin]

TASK [Actualizar paquetes en el sistema] *****
changed: [phpmyadmin]
changed: [apache]

PLAY RECAP *****
apache      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
phpmyadmin  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

Al ejecutar el playbook realizar las siguientes tareas.

- Gathering Facts: Recopila hechos/información de los contenedores remotos como la versión del sistema operativo, IP, versión de Python antes de ejecutar las tareas del playbook.
- Ejecuta el módulo `ansible.builtin.ping`, que realiza ping a los contenedores para ver si están accesibles.
- Ejecuta el módulo `ansible.builtin.apt`, para realizar el *update* y *upgrade* de los contenedores.
- Finalmente da una recopilación de las tareas del playbook, donde indica que la tarea “Actualizar paquetes del sistema” si ha realizado cambios en los contenedores remotos, mientras que la tarea “Ejecutar ping a contenedor” ha resultado satisfactoria pero no ha modificado ningún contenedor.

## 9. Conclusiones y recomendaciones

El objetivo global del proyecto es la entrega y puesta en marcha de una aplicación web siguiendo las buenas prácticas DevOps y utilizando parte de las múltiples herramientas que recomienda la metodología DevOps. Para ello se intentan reflejar las tareas de los equipos que intervienen en la entrega de software, por un lado, el equipo de desarrollo con la función principal de crear software y desarrollar nuevas funcionalidades y mejoras y, por otro lado, el equipo de operaciones con las funciones de implementar, configurar y mantener la infraestructura de TI necesaria para ejecutar y respaldar las aplicaciones desarrolladas.

El objetivo de las buenas prácticas y ciclo de vida DevOps es conseguir la entrega de valor continua de una manera rápida y con la mejor calidad. Esto es posible gracias a un proceso donde interactúan permanentemente los equipos de desarrollo y operaciones.

En el caso de este proyecto por parte del equipo de desarrollo, se ha conseguido por un lado el desarrollo de una sencilla aplicación web -el proyecto se basa mayormente en las tareas del equipo de operaciones y no en las del equipo de desarrollo- utilizando HTML lenguaje del lado cliente y PHP como lenguaje del lado servidor para la recopilación de los datos e inserción de estos en la base de datos MySQL.

Por parte de operaciones, se han conseguido los objetivos principales de un equipo de operaciones dentro del mundo DevOps:

- **Despliegue de la infraestructura TI a través de código** en contenedores docker para la puesta en marcha, orquestación y entrega de los servicios necesarios para el funcionamiento de la aplicación web. Creando un entorno de desarrollo completo con diferentes componentes, como el frontend PHP, la base de datos MySQL y PhpMyAdmin para la gestión de la base de datos.
- El **uso de docker y docker compose** ha permitido la implementación de esta infraestructura TI de forma óptima para la automatización de la puesta en producción de nuevas versiones de la aplicación que incluyan mejoras o nuevas funcionalidades y mantenimiento de la aplicación.
- **Persistencia de datos:** La utilización de volúmenes Docker ha permitido la persistencia de datos importantes, como los datos de la base de datos MySQL y los archivos de configuración SSH, lo que asegura la disponibilidad y la integridad de los datos incluso después de reiniciar los contenedores.
- **Acceso seguro a los contenedores:** La configuración de volúmenes relacionados con la conexión SSH ha facilitado la gestión y administración de los contenedores mediante herramientas como Ansible.
- **Portabilidad del entorno:** La disponibilidad de imágenes Docker en un repositorio público como Docker Hub facilitará la portabilidad del entorno en otros sistemas. Esto permite que el entorno de desarrollo pueda ser replicado fácilmente en diferentes sistemas, garantizando consistencia y uniformidad en el proceso de desarrollo y despliegue de aplicaciones.

- **Gestión y automatización de tareas con Ansible:** La integración de Ansible ha mejorado significativamente la eficiencia operativa al permitir la gestión y automatización de tareas relacionadas con la configuración y el mantenimiento del entorno de contenedores Docker. Quizás en el entorno de este proyecto no se aprecie la capacidad real de Ansible, pero imaginemos un entorno de producción real donde exista un elevado número de contenedores a gestionar y administrar. El uso de Ansible simplifica las operaciones y reduce la posibilidad de errores manuales ofreciendo mayor control y flexibilidad sobre entornos de contenedores.

En conclusión, en este proyecto, mediante la utilización de contenedores Docker, que pueden ser rápidamente replicados y distribuidos en diferentes sistemas TI y Ansible que automatiza la configuración y gestión de la infraestructura, se han podido reflejar las ventajas de la automatización y optimización en la entrega de software siguiendo las prácticas y herramientas DevOps.

## 9.1. Líneas futuras y recomendaciones

Existen muchas posibilidades de mejora dentro del proyecto, que no se han podido abordar principalmente por dos motivos. Falta de tiempo para el estudio de nuevas tecnologías DevOps y falta de tiempo para desarrollarlas dentro del proyecto una vez estudiadas. Las principales mejoras, siempre centradas en la parte de operaciones -ya que es evidente que la parte de desarrollo es muy mejorable y al no ser el objeto de este proyecto no nos vamos a centrar en ellas- son las siguientes:

**Implementación de Kubernetes,** haber realizado la orquestación del entorno de contenedores Docker con una infraestructura basada en Kubernetes. Kubernetes ofrece características avanzadas de orquestación y gestión de contenedores, como escalabilidad automática, balanceo de carga y gestión avanzada de recursos, lo que podría mejorar la escalabilidad y la gestión del entorno en comparación con Docker Compose.

Habiendo utilizado Kubernetes se habrían definido réplicas de los contenedores lo que garantiza la **alta disponibilidad** de los microservicios y por tanto de la aplicación. Definiendo el número de réplicas y gestionando Kubernetes el **balanceo**, el escalado y disminución de los contenedores para satisfacer las variaciones en la demanda de tráfico de la aplicación y mejorando el rendimiento y garantizando el uso eficiente de los recursos.

Otra mejora utilizando Kubernetes es haber creado **dos entornos para desarrollo y producción**, cada uno con su propia configuración de red. Lo que permitiría un aislamiento entre los diferentes entornos garantizando que los cambios realizados en uno no afecten a los otros. Generando políticas de acceso y seguridad específicas para cada entorno, lo que garantiza que solo los usuarios autorizados en ese entorno tengan acceso a los recursos correspondientes.



**Monitorización y registro,** Integrar herramientas de monitorización y registro, como *Prometheus* y *Grafana*, para supervisar el rendimiento y la disponibilidad de los contenedores y servicios, y para facilitar el diagnóstico de problemas y la optimización del rendimiento.

Otra mejora posible en lo que refiere al apartado 6.6 donde recreamos el escenario en otro sistema informático descargando las imágenes desde Docker Hub. En este apartado podría haber lanzado los contenedores con el comando *docker run* añadiendo la opción *-v* para montar los volúmenes al igual que en el entorno de Docker Compose, y tener todos los archivos necesarios para el pleno funcionamiento de la aplicación en un **repositorio *Git* alojado en *GitHub***. De esta manera se hubiese plasmado en el proyecto de manera más clara la portabilidad del entorno de la aplicación utilizando tecnologías habituales en DevOps.

Una última mejora importante por realizar es definir usuarios con permisos específicos para cada tarea, es decir, **limitar el uso del usuario root**. En este proyecto se ha utilizado el usuario root para todas las tareas realizadas por facilidad a la hora de montar el escenario y evitar posibles problemas de permisos que se pudiesen haber presentado y que hubiesen retrasado el proyecto. Por tanto, una tarea creo que muy importante de haber tenido más tiempo es el haber creado usuarios específicos para las distintas tareas, con el fin ofrecer una menor exposición a ataques ya que el usuario root proporciona acceso total al sistema y seguimiento de auditorías y responsabilidad. Por ejemplo, un usuario específico para ejecutar Docker Compose y administrar los contenedores, y otro usuario para ejecutar tareas de Ansible.

## 10. Referencias Bibliográficas

- [1] Oracle VirtualBox Manual de usuario <https://download.virtualbox.org/virtualbox/7.0.6/UserManual.pdf> Fecha del último acceso 02 de marzo de 2024.
- [2] Página de descarga Ubuntu <https://ubuntu.com/download/desktop> Fecha del último acceso 02 de marzo de 2024.
- [3] Página de descarga Visual Studio Code <https://code.visualstudio.com/> Fecha del último acceso 05 de marzo de 2024.
- [4] Documentación oficial de Docker. Disponible en <https://docs.docker.com/> Fecha del último acceso 4 de mayo de 2024.
- [5] Docker Hub. Disponible en <https://hub.docker.com/> Fecha del último acceso 04 de mayo de 2024.
- [6] Manejo de errores de conexión a la base de datos <https://amizbaphp.wordpress.com/2021/01/20/34-manejo-de-errores-con-la-conexion-a-la-base-de-datos/> Fecha del último acceso 11 de marzo de 2024.
- [7] Stackoverflow ¿Cómo puedo agregar la extensión Mysqli o Mysqlnd a mi contenedor docker con php 8.0? <https://es.stackoverflow.com/questions/512287/c%C3%B3mo-puedo-agregar-la-extensi%C3%B3n-mysqli-o-mysqlnd-a-mi-contenedor-docker-con-php> Fecha del último acceso 11 de marzo de 2024.
- [8] Stack Overflow. Cómo habilitar la extensión de PHP en un contenedor Docker <https://stackoverflow.com/questions/57183109/how-to-enable-phps-mysql-extensions-in-docker-container> Fecha del último acceso 8 de abril de 2024.
- [9] Documentación oficial de Ansible <https://docs.ansible.com/> Fecha del último acceso 08 de abril de 2024.

### 10.1. Cursos realizados

- [1] DevOps con Docker, Jenkins, Kubernetes, git, GitFlow CI y CD.  
<https://www.udemy.com/course/devops-con-dockers-kubernetes-jenkins-y-gitflow-cicd/?couponCode=ST13MT40224>
- [2] Aprende Docker, Compose y Swarm.  
<https://www.udemy.com/course/aprende-docker-desde-cero/?couponCode=ST13MT40224>

[3] Ansible desde Cero (2023).

<https://www.udemy.com/course/ansible-desde-cero/?kw=Ansible+desde&src=sac&couponCode=ST13MT40224>