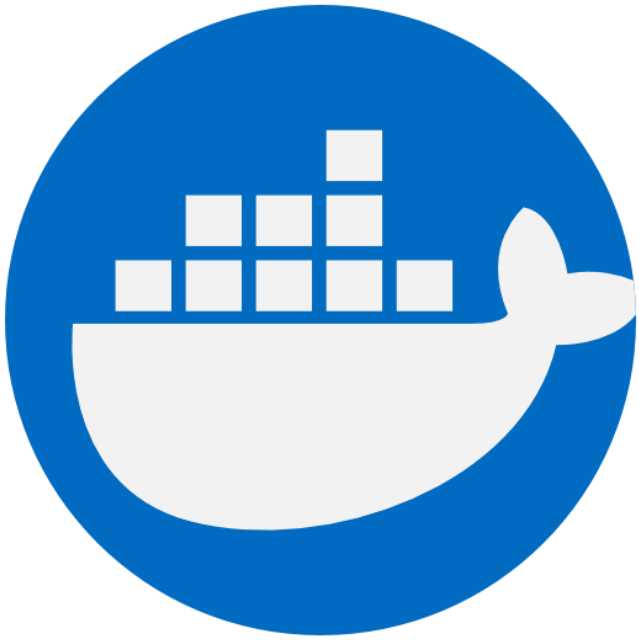
unidad didáctica 01

Docker - Manual de Supervivencia



1. instalar Docker

1.1. introducción

Docker es una aplicación que simplifica el proceso de administración de procesos de aplicación en *contenedores*. Los contenedores le permiten ejecutar sus aplicaciones en procesos con aislamiento de recursos. Son similares a las máquinas virtuales, pero los contenedores son más portátiles, más flexibles con los recursos y más dependientes del sistema operativo host.

Para hallar una introducción detallada a los distintos componentes de un contenedor de Docker, consulte El ecosistema de Docker: Introducción a los componentes comunes.

En este tutorial, instalará y usará Docker Community Edition (CE) en Ubuntu 20.04. Instalará Docker, trabajará con contenedores e imágenes e introducirá una imagen en un repositorio de Docker.

1.2. paso 1: Instalar Docker

Es posible que la versión del paquete de instalación de Docker disponible en el repositorio oficial de Ubuntu no sea la más reciente. Para asegurarnos de contar con la versión más reciente, instalaremos Docker desde el repositorio oficial de Docker. Para hacerlo, agregaremos una nueva fuente de paquetes y la clave GPG de Docker para garantizar que las descargas sean válidas, y luego instalaremos el paquete.

1. Primero, actualiza tu lista de paquetes existente:

```
$ sudo apt update
```

2. A continuación, instala algunos paquetes de requisitos previos que permitan a `apt` usar paquetes a través de HTTPS:

```
$ sudo apt install apt-transport-https ca-certificates curl software-
properties-common
```

3. Luego, añade la clave de GPG para el repositorio oficial de Docker en su sistema:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Agrega el repositorio de Docker a las fuentes de APT:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

5. A continuación, actualiza el paquete de base de datos con los paquetes de Docker del repositorio recién agregado:

```
$ sudo apt update
```

6. Asegúrate de estar a punto de realizar la instalación desde el repositorio de Docker en lugar del repositorio predeterminado de Ubuntu:

```
$ apt-cache policy docker-ce
```

Si bien el número de versión de Docker puede ser distinto, verás un resultado como el siguiente:

`Output of apt-cache policy docker-ce`

```
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.9~3-0~ubuntu-focal
  Version table:
     5:19.03.9~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64
Packages
```

Observa que `docker-ce` no está instalado, pero la opción más viable para la instalación es del repositorio de Docker para Ubuntu 20.04 (`focal`).

7. Por último, instala Docker:

```
sudo apt install docker-ce
```

Con esto, Docker quedará instalado, el demonio se iniciará y el proceso se habilitará para ejecutarse en el inicio.

8. Comprueba que funcione:

```
sudo systemctl status docker
```

El resultado debe ser similar al siguiente, y mostrar que el servicio está activo y en ejecución:

```
Output
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 24321 (dockerd)
      Tasks: 8
     Memory: 46.4M
     CGroup: /system.slice/docker.service
             └─24321 /usr/bin/dockerd -H fd:// --
containerd=/run/containerd/containerd.sock
```

La instalación de Docker ahora te proporcionará no solo el servicio de Docker (demonio) sino también la utilidad de línea de comandos `docker` o el cliente de Docker.

2. paso 2: Instalar Docker Compose

2.1. instalar Docker Compose

Para asegurarnos de que obtenemos la versión estable más reciente de Docker Compose, descargaremos este software de su repositorio oficial de Github.

1. Primero, confirmamos la versión más reciente disponible en su página de versiones.

El siguiente comando descargará la última versión y guardará el archivo ejecutable en `/usr/local/bin/docker-compose`, que hará que este software esté globalmente accesible como `docker-compose`:

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.26.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. A continuación, estableceremos los permisos correctos para que el comando `docker-compose` sea ejecutable:

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. Para verificar que la instalación se realizó correctamente, puedes ejecutar:

```
docker-compose --version
```

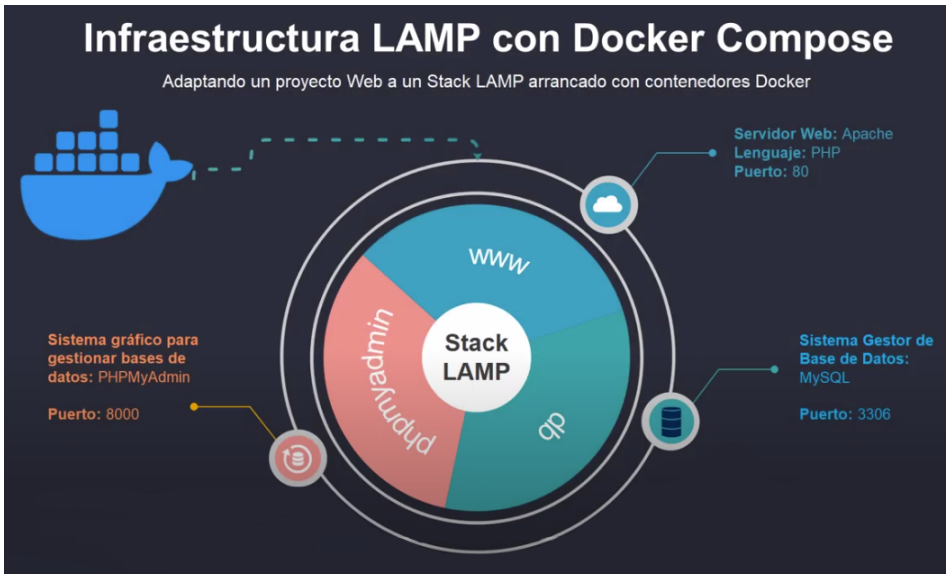
Visualizarás un resultado similar a esto:

```
Output
docker-compose version 1.26.0, build 8a1c60f6
```

Docker Compose se ha instalado correctamente en tu sistema. A continuación, veremos cómo configurar un archivo `docker-compose.yml` y obtener un entorno en contenedor listo para usarse con esta herramienta.

2.2. configurar un archivo `docker-compose.yml`

Para demostrar cómo configurar un archivo `docker-compose-yml` y trabajar con Docker Compose, crearemos un entorno de servidor web Apache, PHP, MySQL y PHPMyAdmin (LAMP). Este entorno en contenedor servirá como web dinámica; lo que nos servirá en un futuro para acceder a bases de datos en este módulo.



1. Comienza creando un nuevo directorio en tu carpeta de inicio, y luego muévelo a él:

```
mkdir ~/pruNombreEstudiante
cd ~/pruNombreEstudiante
```

2. En este directorio, configura una carpeta de aplicaciones que servirá como la raíz del documento para tu entorno:

```
mkdir miProyecto
```

3. Copia el sistema de carpetas (ofrecido por el profesor) dentro de la carpeta anterior:
4. A continuación, abre el proyecto con Visual Studio Code:

```
code .
```

5. Abre el archivo `docker-compose.yml` :

```
version: "3.1"
services:
    db:
        image: mysql
        ports:
            - "3306:3306"
        command: --default-authentication-plugin=mysql_native_password
        environment:
            MYSQL_DATABASE: pruDB
            MYSQL_PASSWORD: dwes
            MYSQL_ROOT_PASSWORD: dwes
        volumes:
            - ./dump:/docker-entrypoint-initdb.d
            - ./conf:/etc/mysql/conf.d
            - persistent:/var/lib/mysql
        networks:
            - default
    www:
        build: .
        ports:
```

```
            - "80:80"
        volumes:
            - ./www:/var/www/html
        links:
            - db
        networks:
            - default
    phpmyadmin:
        image: phpmyadmin/phpmyadmin
        links:
            - db:db
        ports:
            - 8000:80
        environment:
            MYSQL_USER: root
            MYSQL_PASSWORD: dwes
            MYSQL_ROOT_PASSWORD: dwes
volumes:
    persistent:
```

El archivo `docker-compose.yml` normalmente comienza con la definición de `la versión`. Esto indicará a Docker Compose qué versión de la configuración estamos usando.

Luego tenemos el bloque `services`, donde configuramos los servicios que son parte de este entorno. En nuestro caso, tenemos un servicio llamado `www`, otro `db` y otro `phpmyadmin`.

El servicio `www` utiliza la imagen `.` y establece una redirección de puerto con la directiva `ports`. Todas las solicitudes en el puerto `8000` del equipo **host** (el sistema desde el cual está ejecutando Docker Compose) serán redirigidas al contenedor `www` en el puerto `80`.

La directiva `volumes` creará un volumen compartido entre el equipo host y el contenedor. Esto compartirá la carpeta `www` local con el contenedor, y el volumen se ubicará en `/var/www/html` dentro del contenedor, que luego sobreescribirá la raíz predeterminada del documento para Apache.

Hemos creado una página demo `index.php` y un archivo `docker-compose.yml` para crear un entorno de servidor web en el contenedor que lo presentará. En el siguiente paso, abriremos este entorno con Docker Compose.

2.3. ejecutar Docker Compose

Con el archivo `docker-compose.yml` implementado, podemos ejecutar Docker Compose para mostrar nuestro entorno. El siguiente comando descargará las imágenes Docker necesarias, creará un contenedor para el servicio `web` y ejecutará el entorno en contenedor en modo segundo plano:

```
docker-compose up -d
```

Docker Compose primero buscará la imagen definida en su sistema local, y si no puede encontrar la imagen, descargará la imagen desde Docker Hub. Verás un resultado como este:

```
Output
Creating network "docker-lamp-main2_default" with the default driver
Creating volume "docker-lamp-main2_persistent" with default driver
```

```
Building www
[+] Building 2.4s (11/11) FINISHED                                ocker:default
 => [internal] load build definition from Dockerfile                       0.0s
 => => transferring dockerfile: 611B                                       0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load metadata for docker.io/library/php:8.0.0-apache        2.3s
 => [1/7] FROM docker.io/library/php:8.0.0-apache@sha256:d99ca98b9fe768 0.0s
 => CACHED [2/7] RUN docker-php-ext-install mysqli                         0.0s
 => CACHED [3/7] RUN apt-get update     && apt-get install -y sendmail
libpng-dev     && apt-get install -y libzip-dev     && apt-get install -y
zlib1g-dev   && apt-get install -y 0.0s
 => CACHED [4/7] RUN docker-php-ext-install mbstring                       0.0s
 => CACHED [5/7] RUN docker-php-ext-install zip                            0.0s
 => CACHED [6/7] RUN docker-php-ext-install gd                             0.0s
 => CACHED [7/7] RUN a2enmod rewrite                                       0.0s
 => exporting to image                                                     0.0s
 => => exporting layers                                                    0.0s
 => => writing image sha256:a35d05b133deb154d2188bf33e3                    0.0s
 => => naming to docker.io/library/docker-lamp-main2_www                   0.0s
WARNING: Image for service www was built because it did not already exist. To
rebuild this image you must use `docker-compose build` or `docker-compose up
--build`.
Creating docker-lamp-main2_db_1 ... done
Creating docker-lamp-main2_www_1        ... done
Creating docker-lamp-main2_phpmyadmin_1 ... done
```

Tu entorno ahora está funcionando en segundo plano. Para verificar que el contenedor está activo, puede ejecutar:

```
docker-compose ps
```

Este comando le mostrará información sobre los contenedores en ejecución y su estado, además de cualquier redireccionamiento de puertos en vigor actualmente:

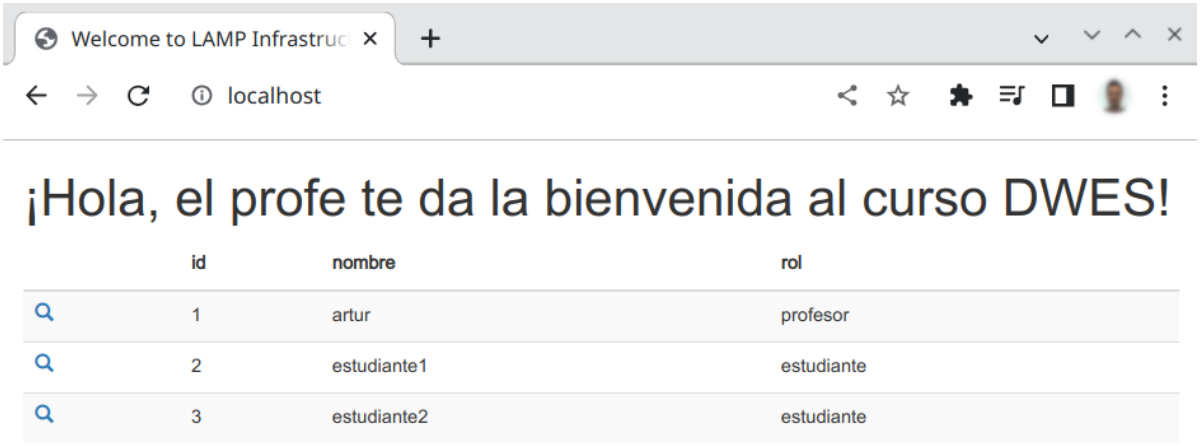
```
Output
        Name                      Command                  State       Ports

-------------------------------------------------------------------------------
---------
docker-lamp-main2_db_1       docker-entrypoint.sh --def ...   Exit      1

docker-lamp-main2_phpmyadmin_1 /docker-entrypoint.sh apac ... Up
0.0.0.0:8000->80/tcp,:::8000->80/tcp
docker-lamp-main2_www_1      docker-php-entrypoint apac ...   Up
0.0.0.0:80->80/tcp,:::80->80/tcp
```

Ahora puedes acceder a la aplicación demo apuntando tu servidor web a `localhost:80` si estás ejecutando esta demo en tu equipo local, o a `:8000` si estás ejecutando esta demo en un servidor remoto.

Verás una página como la siguiente:



El volumen compartido que ha configurado en el archivo `docker-compose.yml` mantiene los archivos de tu carpeta `www` sincronizados con la raíz del documento del contenedor. Si realizas algún cambio al archivo `index.php`, serán recogidos automáicamente por el contenedor y se reflejarán en tu navegador cuando vuelva a cargar la página.

En el siguiente paso, verás cómo gestionar tu entorno en contenedor con los comandos de Docker Compose.

2.4. parar y Eliminar imagen base

Cuando tienes **contenedores Docker** ejecutándose, primero necesitas detenerlos antes de borrarlos.

1. Detén todos los *contenedores* ejecutándose:

```
$ sudo docker stop $(docker ps -a -q)
```

2. Elimina todos los *contenedores* detenidos:

```
$ sudo docker rm $(docker ps -a -q)
```

2.4.1. limpieza del entorno Docker: ¡prune!

Este comando nos ahorra la eliminación manual de cada recurso permitiéndonos hacer una **limpieza general** del entorno rápidamente.

Lo podemos utilizar de varias maneras:

1. Elimina todos los contenedores **frenados** y redes no utilizadas. También elimina las imágenes temporales.

```
$ sudo docker system prune
```

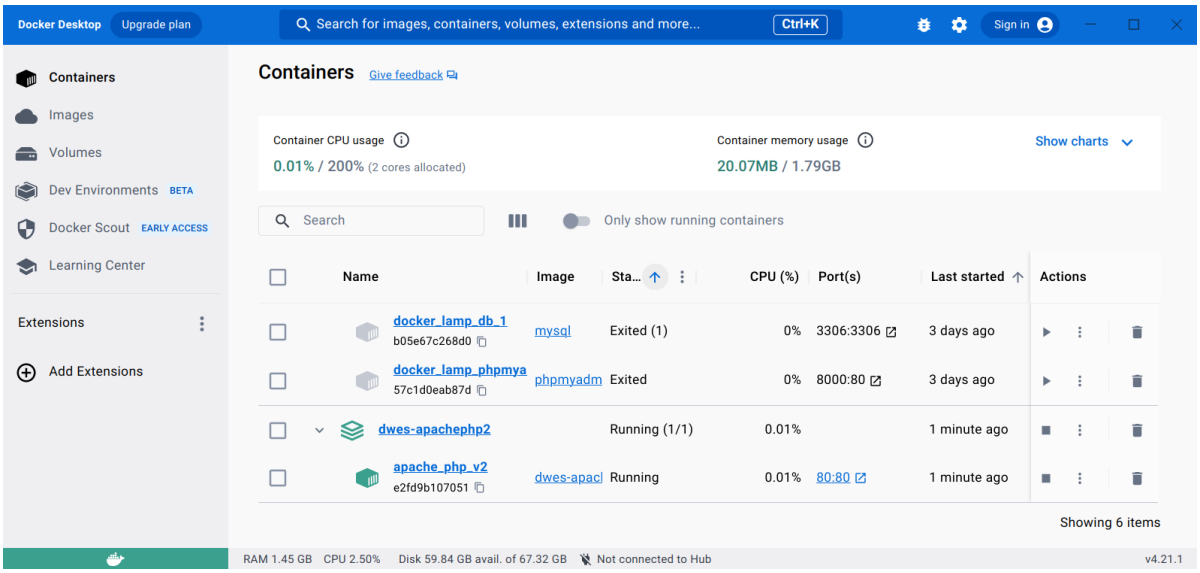
2. Elimina todas las imágenes **no utilizadas por algún contenedor**.

```
$ sudo docker system prune -a
```

3. paso 3. Instalar Docker Desktop

Con esta aplicación será la forma más sencilla de ejecutar, compilar, depurar y probar las aplicaciones Dockerized.

El escritorio Docker consta de herramientas para desarrolladores, Aplicación Docker, Kubernetes y sincronización de versiones. Nos permite crear imágenes y plantillas certificadas de nuestra elección de idiomas y herramientas.



4. paso 4. Instalar Docker Portainer

ortainer es una **forma cómoda de gestionar entornos de contenedores distribuidos**.

El software se instala como un contenedor Docker y, por tanto, se ejecuta prácticamente en cualquier lugar. Mostramos la rutina de instalación y aportamos útiles consejos.

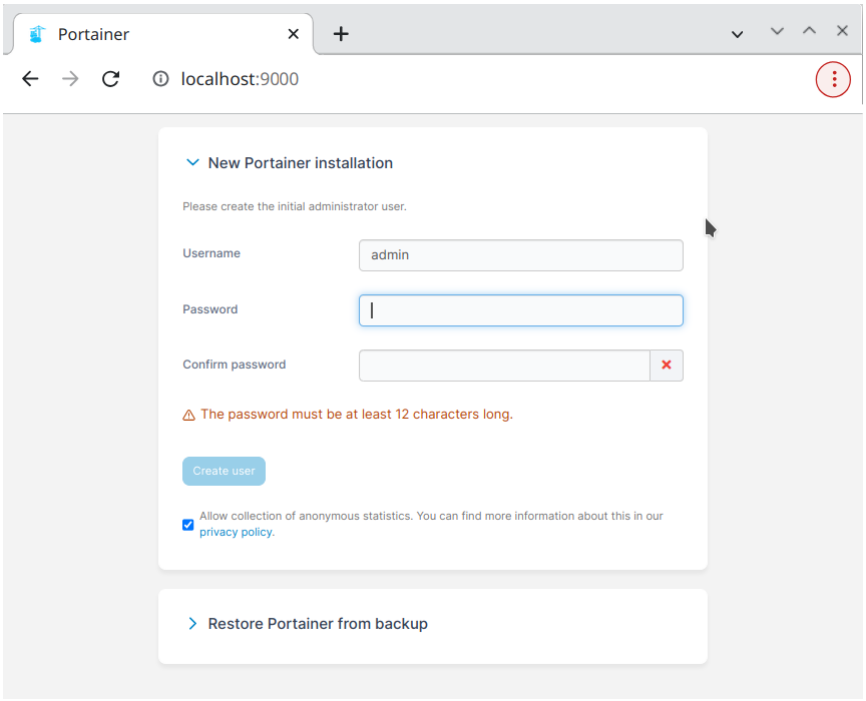
4.1. instalar

```
$ sudo docker volume create portainer_data
$ sudo docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock
-v portainer_data:/data portainer/portainer
```

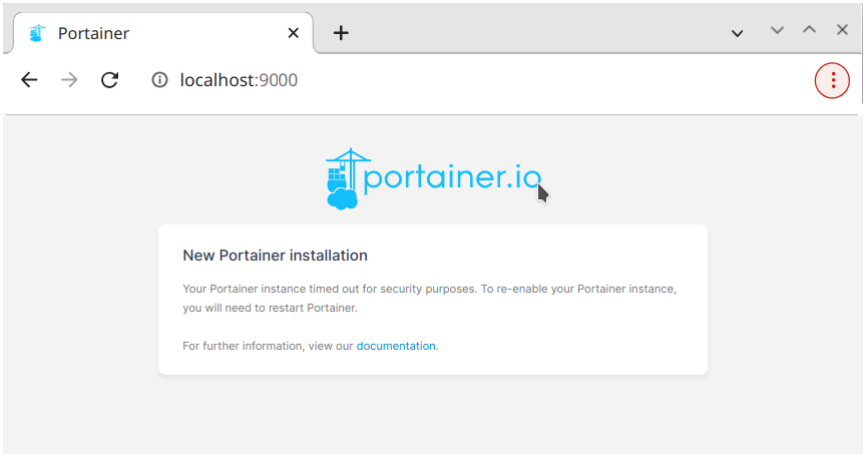
```
abc@abc-pc:~/projectes/dockerlaravel$ sudo docker volume create portainer_data
[sudo] password for abc:
portainer_data
abc@abc-pc:~/projectes/dockerlaravel$ sudo docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
Unable to find image 'portainer/portainer:latest' locally
latest: Pulling from portainer/portainer
772227786281: Pull complete
96fd13befc87: Pull complete
0bad1d247b5b: Pull complete
b5d1b01b1d39: Pull complete
Digest: sha256:47b064434edf437badf7337e516e07f64477485c8ecc663ddabbe824b20c672d
Status: Downloaded newer image for portainer/portainer:latest
40023d727946867b8db71f8ca3aa03d9a988b7f6efeafd8885f73023675077cc
```

4.2. lanzar

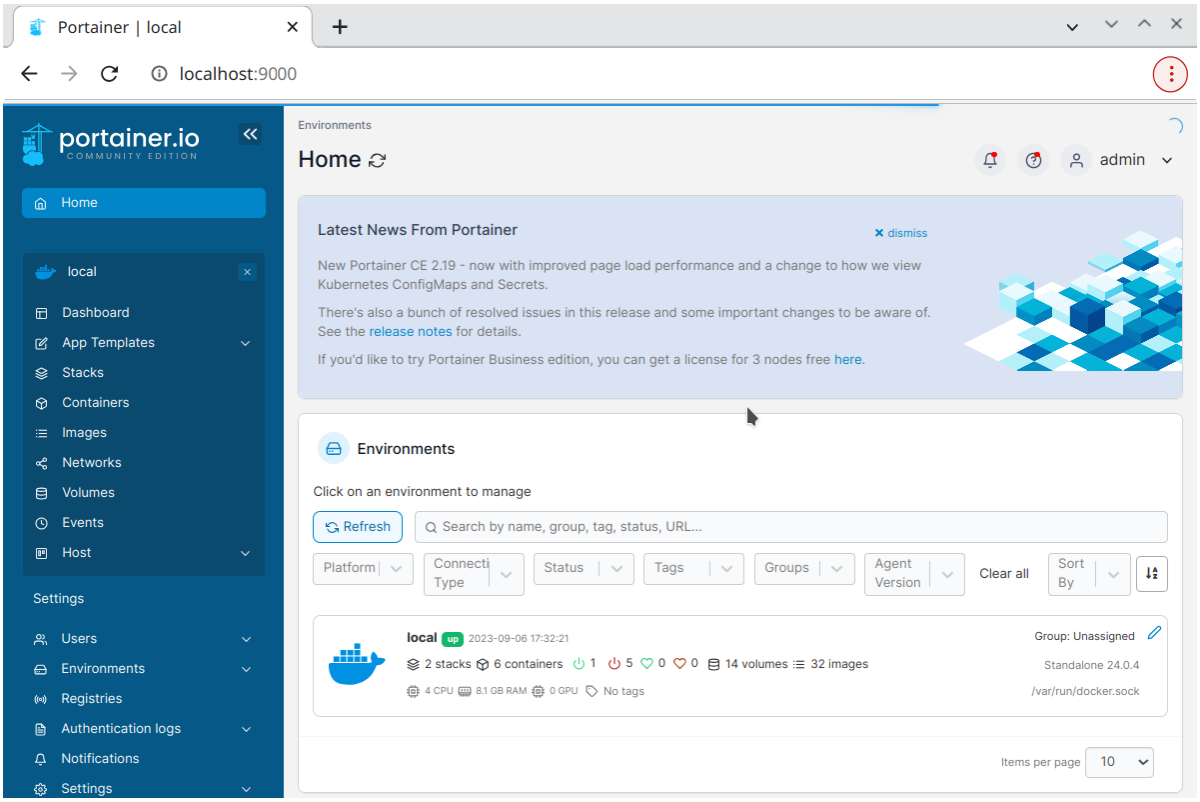
Como otro contenedor, podremos acceder a él mediante un navegador (Firefox, Chrome, ...) introduciendo la URL: **localhost:9000**



Si muestra el siguiente texto, debemos reiniciar el contenedor del Docker Portainer:



Volvemos a acceder vía **localhost:9000**:



4.3. funciona como un docker

Para ver el ID de contenedor podemos listar los contenedores:

```
$ sudo docker ps -a
```

Para parar el contenedor:

```
$ sudo docker stop [ID]
```

Para volver a lanzar el contenedor:

```
$ sudo docker start [ID]
```

5. bibliografía

- web digitalocean.
- canal Youtube DevOpsea.
- apuntes profesor Aitor Medrano https://aitor-medrano.github.io