Inicio a la Virtualización

En esta serie de documentos encontrarás toda la información que he ido recopilando durante el tiempo que llevo interesado con este tema, ya sea diferencias de Hypervisores, qué tipos de sistemas se virtualizan y programas / sistemas operativos.

- Tipos de Hypervisores
- Tipos de Virtualización
    - Máquina Virtual
    - LXC LinuX Containers
    - Docker
    - Docker Compose

Tipos de Hypervisores

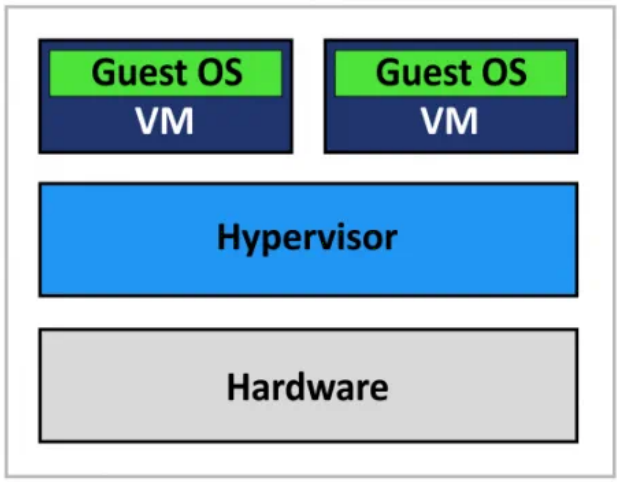
¿Qué es la virtualización?

La virtualización no deja de ser una simulación o emulación de un entorno de software, ya sea un SO o una aplicación o conjunto de aplicaciones. Separando los recursos físicos (hardware) para crear sistemas independientes, los unos de los otros como máquinas virtuales o contenedores (LXC, Docker...). Independientemente de lo que vayamos a virtualizar tenemos que saber que son los “Hipervisores”, los encargados de ejecutar todo tipo de MV o Contenedor y es el encargado de asignar aspectos básicos para el funcionamiento como la CPU, RAM, Almacenamiento entre otros a cada VM o contenedor.

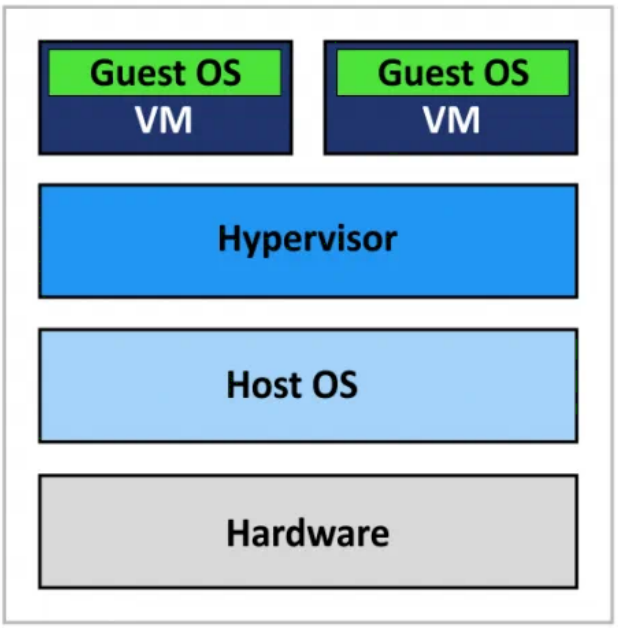
Tipos de hipervisor

> Tipo 1→ Es el que no necesita sistema operativo, sino que se ejecuta en su lugar, lo que reduce considerablemente los recursos que se le dan al host. Con lo que este tipo se suele usar para los servidores, ya que no es necesario tener un sistema operativo debajo del hipervisor que ocupe recursos inútilmente. Ejemplos de este tipo de hipervisor tenemos Hyper-V, ESXi, Proxmox entre otros.

> Tipo 2→ Es el hipervisor ejecutado en un sistema operativo, ya sea “Windows” o “Linux” como si fuera un programa más. Este tipo suele ser el más común, ya que para uso personal es la mejor opción, porque tienes tu sistema operativo host y después tus VM. Ejemplos de este tipo de hipervisor tenemos los dos más populares VMWare y VirtualBox del que vamos a hablar más en profundidad en el resto del documento.



Type 1 Hypervisor
(Bare-Metal Architecture)



Type 2 Hypervisor
(Hosted Architecture)

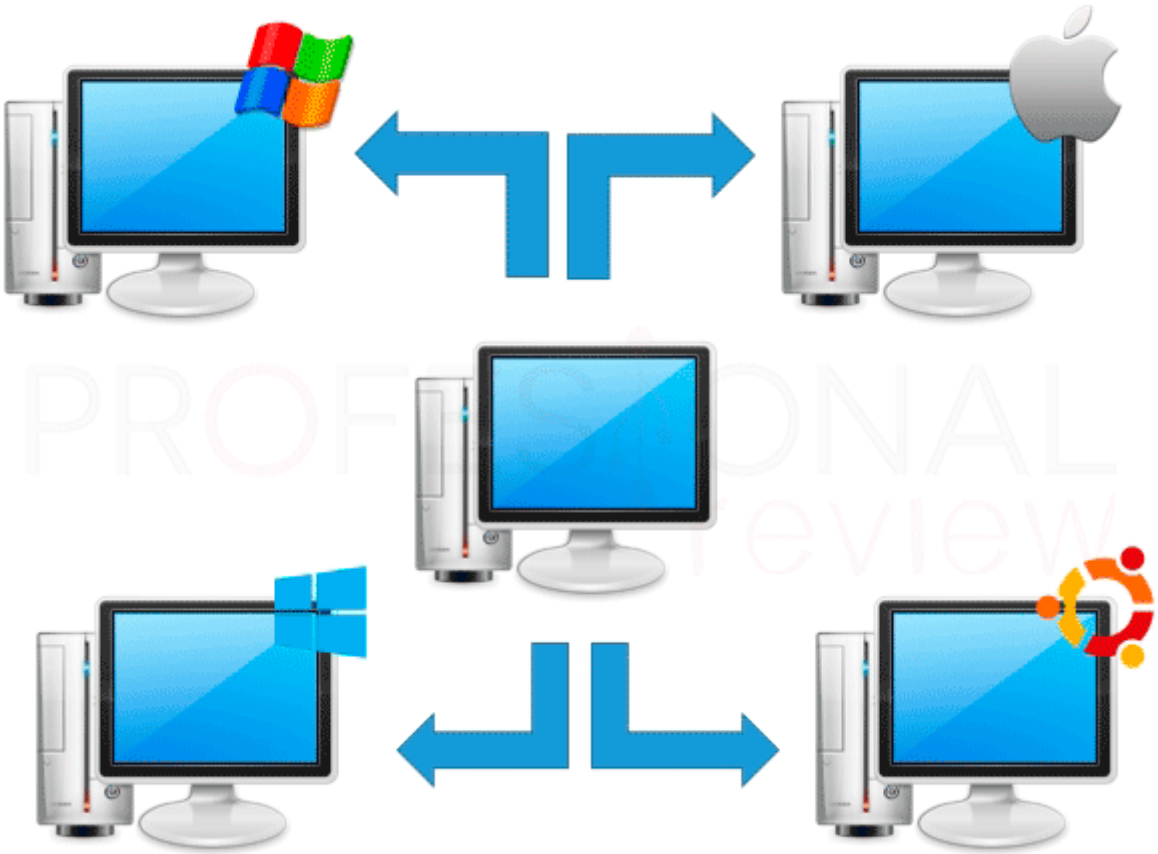
Tipos de Virtualización

Explicación de los tipos más usados de la virtualización, como por ejemplo Máquinas virtuales y Contenedores.

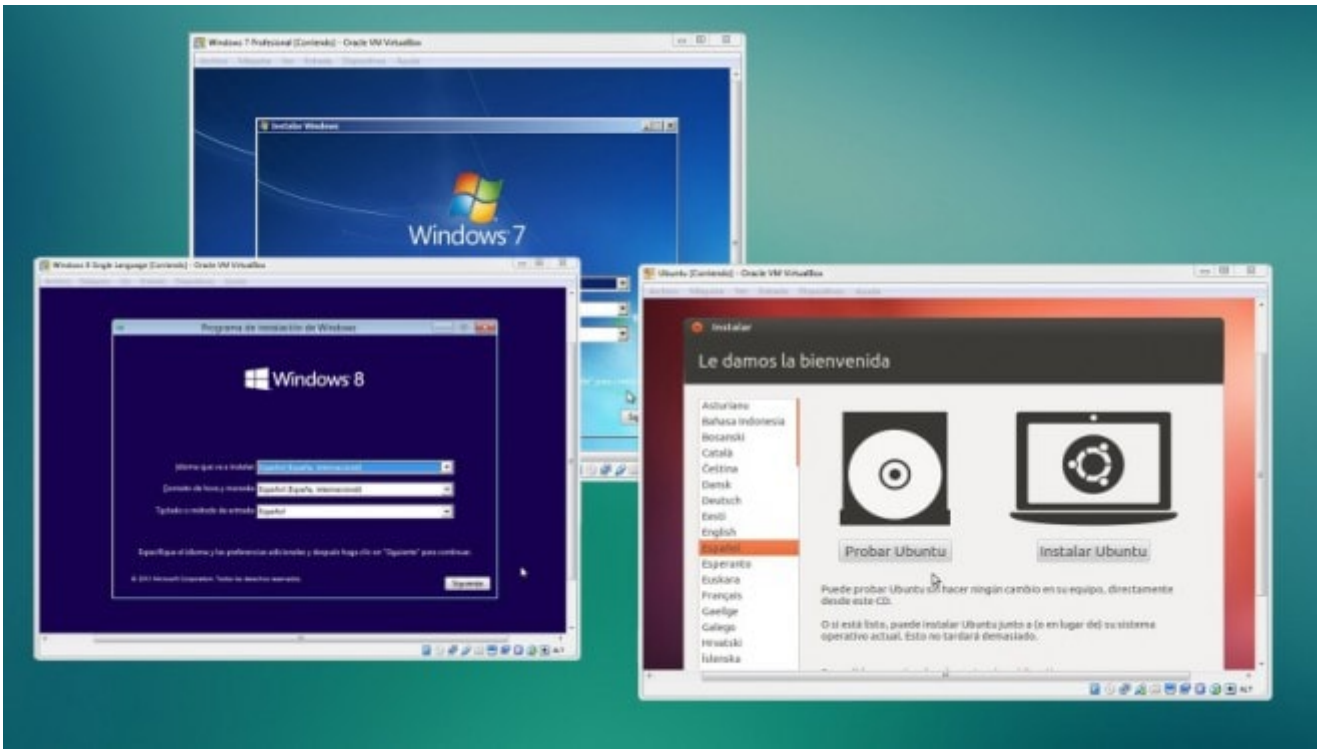
Máquina Virtual

¿Qué es una máquina virtual?

Una máquina virtual es un software que crea una capa independiente donde se emula el funcionamiento de una máquina real con sus respectivos componentes de hardware que necesita para funcionar (disco duro, memoria RAM, tarjetas de red gráfica, etc.), esta puede ejecutar cualquier sistema operativo o programa como lo haría una máquina real.

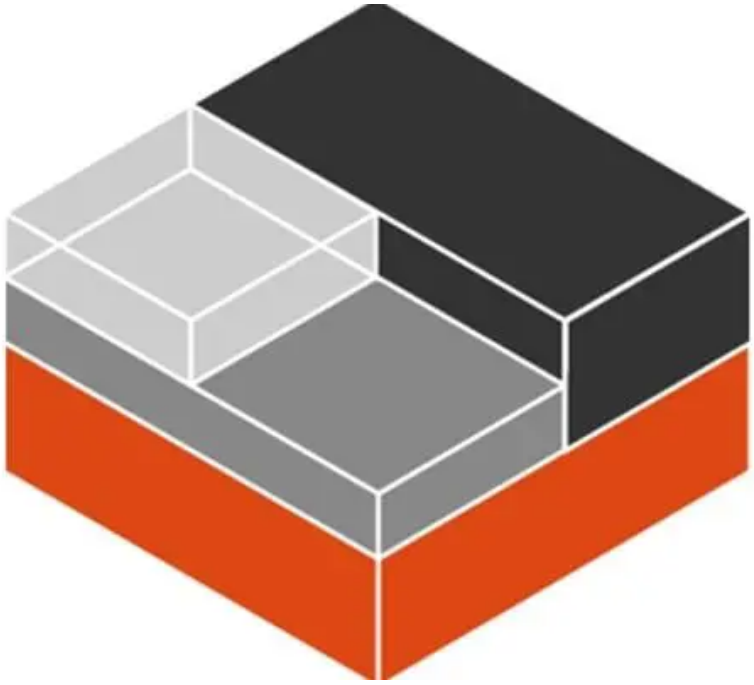


Toda esta simulación se encapsula en una serie de archivos que actúan como contenedor desde el que se ejecutará la máquina virtual en una ventana de nuestro ordenador como cualquier otro programa y sin el peligro de que nada qué le suceda acabe afectando a nuestra máquina real la cual está ejecutándola.


Windows 7
Windows 8
Instalar
Le damos la bienvenida
Probar Ubuntu
Instalar Ubuntu

LXC LinuX Containers

¿Qué es LXC?



LXC o linuX Containers son contenedores aislados con su propia reserva de recursos y se realiza mediante entornos de virtualización. Los contenedores no emulan hardware, así como las máquinas virtuales sí. LXC es un sistema operativo dentro de nuestro sistema y a efectos prácticos se comporta como una máquina virtual, la emulación se ejecuta en el **kernel** de linux. Pero **LXC** proporciona un **contenedor** mínimo, así reduciendo el uso de recursos de nuestro sistema y aumentando la portabilidad gracias a que se pueden crear plantillas que se pueden ejecutar en cualquier contenedor.

Hay dos formas de crearlos

- Por terminal
- Por interfaz gráfica (solo algunos S.O la tienen ej. Proxmox)

Terminal

```
lxc-create -n MyCentOSContainer1 -t /usr/local/share/lxc/templates/lxc-centos/usr/local/share/lxc/templates/lxc-centos
```

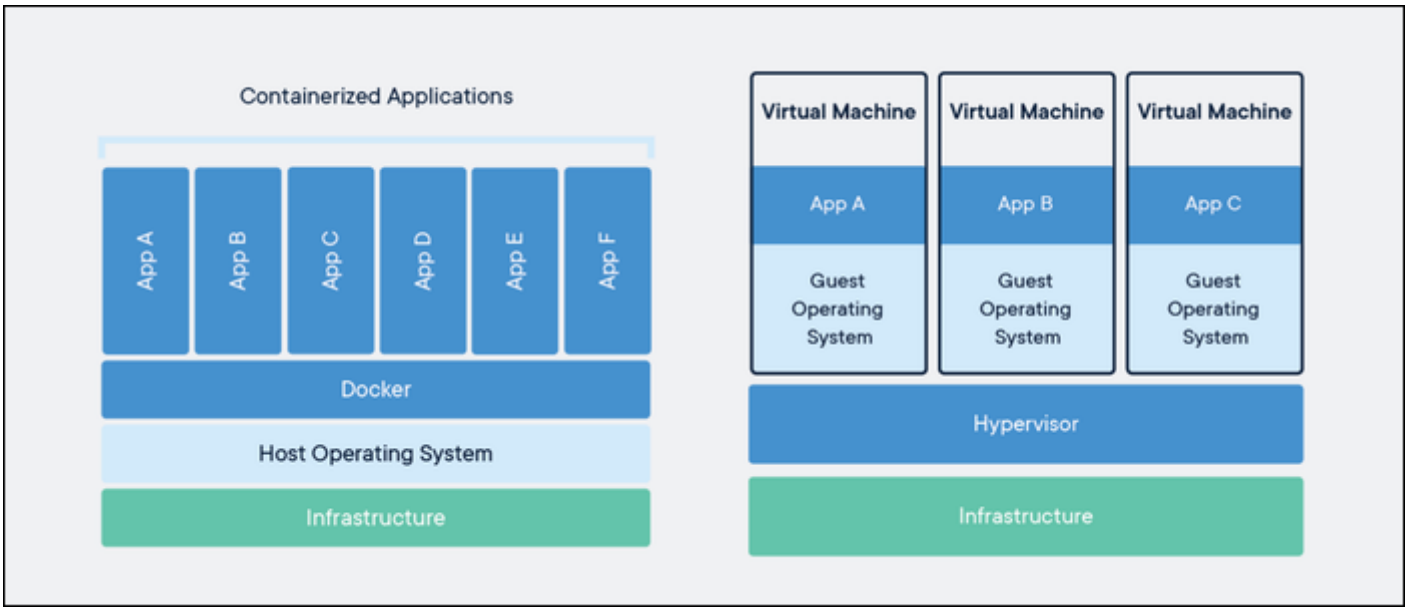
Docker

Docker

La función principal de Docker es: desarrollar, enviar y ejecutar cualquier aplicación en cualquier sistema, constituyéndose así como una alternativa flexible y capaz de ahorrar recursos frente a la emulación de componentes de hardware basada en máquinas virtuales (VM).



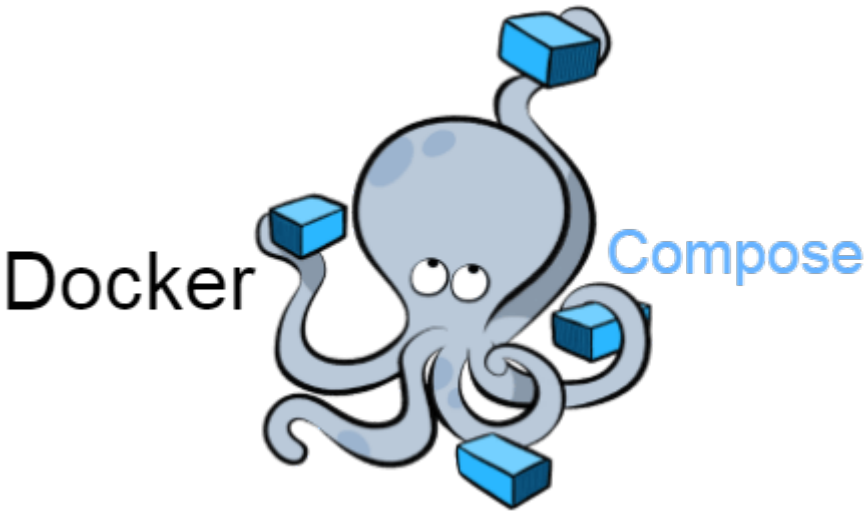
Debido a que la virtualización de hardware tradicional se basa en iniciar diferentes sistemas operativos en el mismo sistema host (host), con la ayuda de Docker, gracias al llamado contenedor, las aplicaciones se ejecutan como procesos aislados en el mismo sistema. Por lo tanto, hablamos de virtualización basada en contenedores y, pues, también hablamos de virtualización a nivel de sistema operativo. Portainer App para automatización de Docker y para su uso más rápido y fácil gracias a su interfaz de usuario en web.



Ejemplo de creación de contenedor Docker

```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
    --restart=always \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v portainer_data:/data \
    portainer/portainer-ce:2.9.3
```

Docker Compose



Docker Compose es una utilidad de Docker que nos ayuda a la hora de crear múltiples contenedores Docker al mismo tiempo. A diferencia de Docker Cli este no hay que crearlo directamente en la terminal, sino que se crea un archivo .yml que es más fácil de ser modificado y configurado que en la propia terminal. Una vez creado solo se tiene que ejecutar un comando por terminal que comprobará el archivo e instalara y configurara lo que le hemos descrito en el mismo.

Creación de WordPress por Docker Cli

Primero se crea el Docker de la base de datos.

```
docker create \
  --name=mariadb \
  --net=lsio \
  -e PUID=1000 \
  -e PGID=1000 \
  -e MYSQL_ROOT_PASSWORD=mariadbpassword \
  -e TZ=Europe/London \
  -e MYSQL_DATABASE=WP_database \
  -e MYSQL_USER=WP_dbuser \
  -e MYSQL_PASSWORD=WP_dbpassword \
  -v /home/aptalca/appdata/mariadb:/config \
  --restart unless-stopped \
```

```
  lscr.io/linuxserver/mariadb
```

Y de la misma forma que la base de datos tendremos que crear el contenedor donde irá instalado la aplicación de WordPress.

```
docker create \
  --name=swag \
  --cap-add=NET_ADMIN \
  --net=lsio \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Europe/London \
  -e URL=linuxserver-test.com \
  -e SUBDOMAINS=wildcard \
  -e VALIDATION=dns \
  -e DNSPLUGIN=cloudflare \
  -p 443:443 \
  -p 80:80 \
  -v /home/aptalca/appdata/swag:/config \
  --restart unless-stopped \
  lscr.io/linuxserver/swag
```

> Una vez creadas los dos contenedores los tendremos que poner en la misma red de docker para que se puedan comunicar. Y pos ultimo dentro del segundo contenedor creado tendermos que instalar Wordpress:

```
wget https://wordpress.org/latest.tar.gz
tar xvf latest.tar.gz -C /home/aptalca/appdata/swag/www/
rm latest.tar.gz
```

Creado con Docker Compose

```
---
version: "2.1"
services:
  mariadb:
    image: lscr.io/linuxserver/mariadb
    container_name: mariadb
    environment:
```

```
      - PUID=1000
      - PGID=1000
      - MYSQL_ROOT_PASSWORD=mariadbpassword
      - TZ=Europe/London
      - MYSQL_DATABASE=WP_database
      - MYSQL_USER=WP_dbuser
      - MYSQL_PASSWORD=WP_dbpassword
    volumes:
      - /home/aptalca/appdata/mariadb:/config
    restart: unless-stopped
  swag:
    image: lscr.io/linuxserver/swag
    container_name: swag
    cap_add:
      - NET_ADMIN
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/London
      - URL=linuxserver-test.com
      - SUBDOMAINS=wildcard
      - VALIDATION=dns
      - DNSPLUGIN=cloudflare
    volumes:
      - /home/aptalca/appdata/swag:/config
    ports:
      - 443:443
      - 80:80
    depends_on:
      - mariadb
    restart: unless-stopped
```

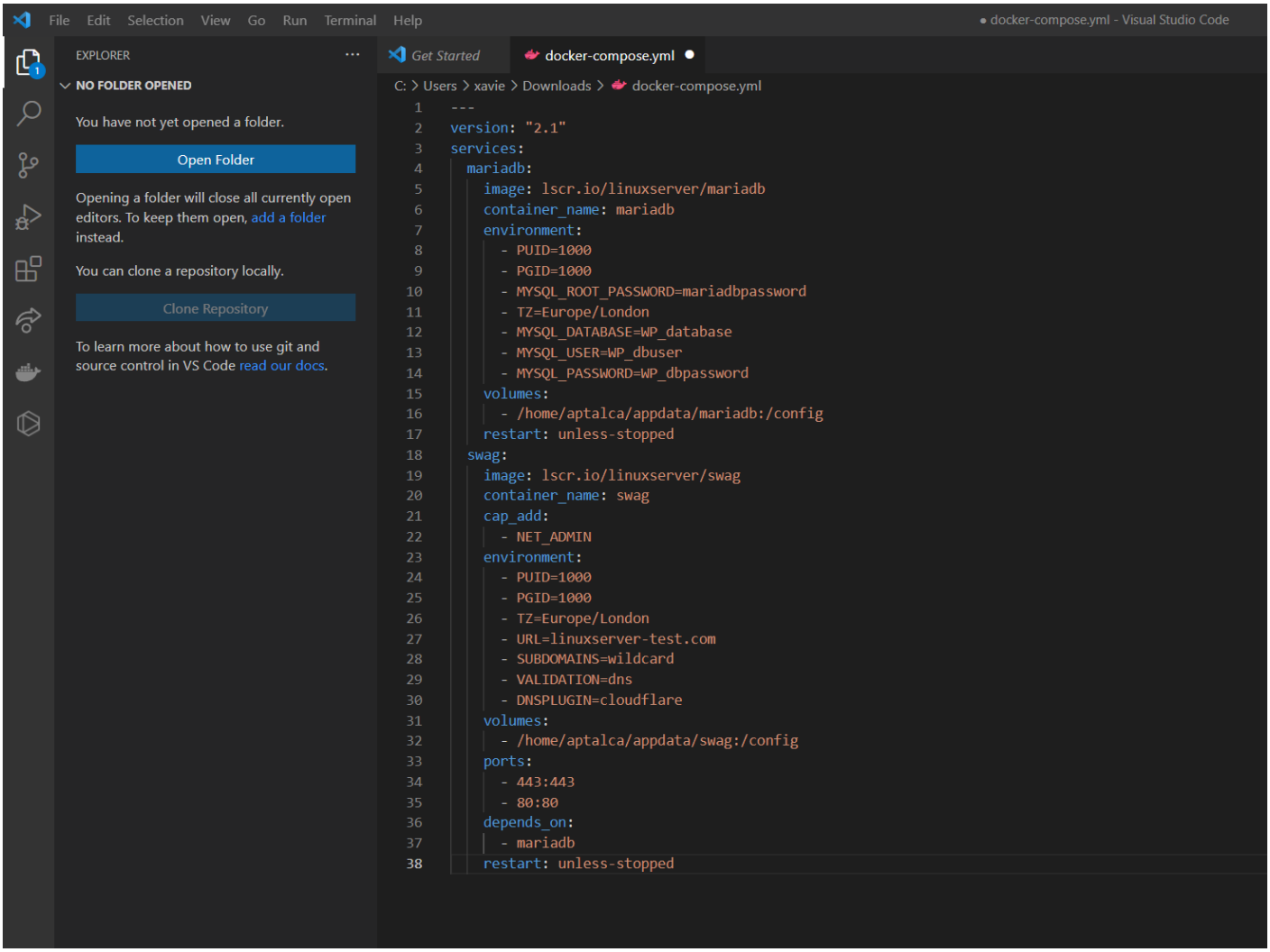
> A diferencia de Docker Cli, en Docker Compose se pude hacer todo en el mismo archivo y despues ejecutarlo en terminal

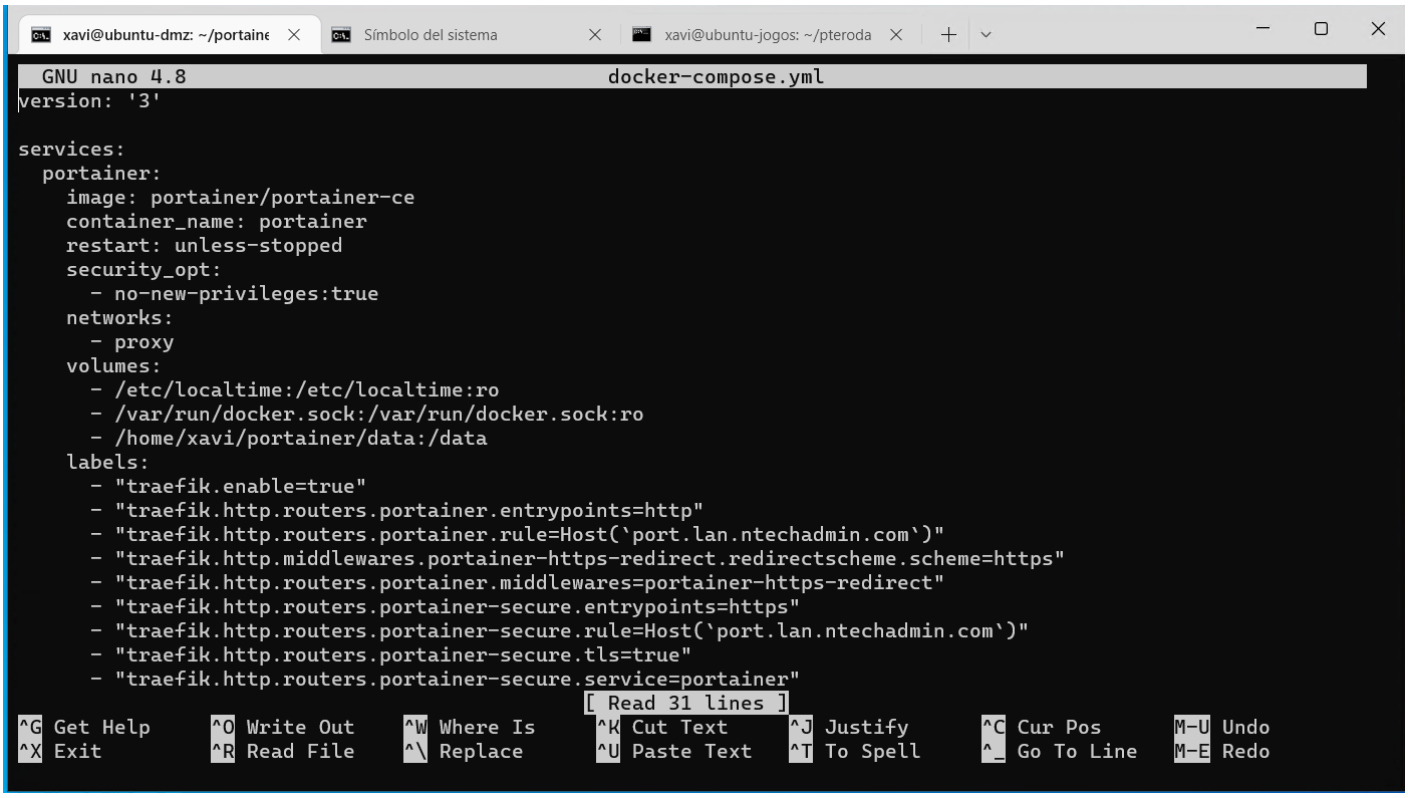
```
docker-compose up -d
```

> Una vez el docker esta creado, si se modifica el archivo se tendra que crear los docker de nuevo y prar eso esta el siguiente comando

```
docker-compose up -d --force-recreate
```

> Puede ser creado y configurado en un editor de texto y despues trasladarlo al archivo .yml de nuestro server.




xavi@ubuntu-dmz: ~/portaine
Símbolo del sistema
xavi@ubuntu-jogos: ~/pteroda
GNU nano 4.8 docker-compose.yml
version: '3'
services:
portainer:
image: portainer/portainer-ce
container_name: portainer
restart: unless-stopped
security_opt:
- no-new-privileges:true
networks:
- proxy
volumes:
- /etc/localtime:/etc/localtime:ro
- /var/run/docker.sock:/var/run/docker.sock:ro
- /home/xavi/portainer/data:/data
labels:
- "traefik.enable=true"
- "traefik.http.routers.portainer.entrypoints=http"
- "traefik.http.routers.portainer.rule=Host(`port.lan.ntechadmin.com`)"
- "traefik.http.middlewares.portainer-https-redirect.redirectscheme.scheme=https"
- "traefik.http.routers.portainer.middlewares=portainer-https-redirect"
- "traefik.http.routers.portainer-secure.entrypoints=https"
- "traefik.http.routers.portainer-secure.rule=Host(`port.lan.ntechadmin.com`)"
- "traefik.http.routers.portainer-secure.tls=true"
- "traefik.http.routers.portainer-secure.service=portainer"
[Read 31 lines]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo