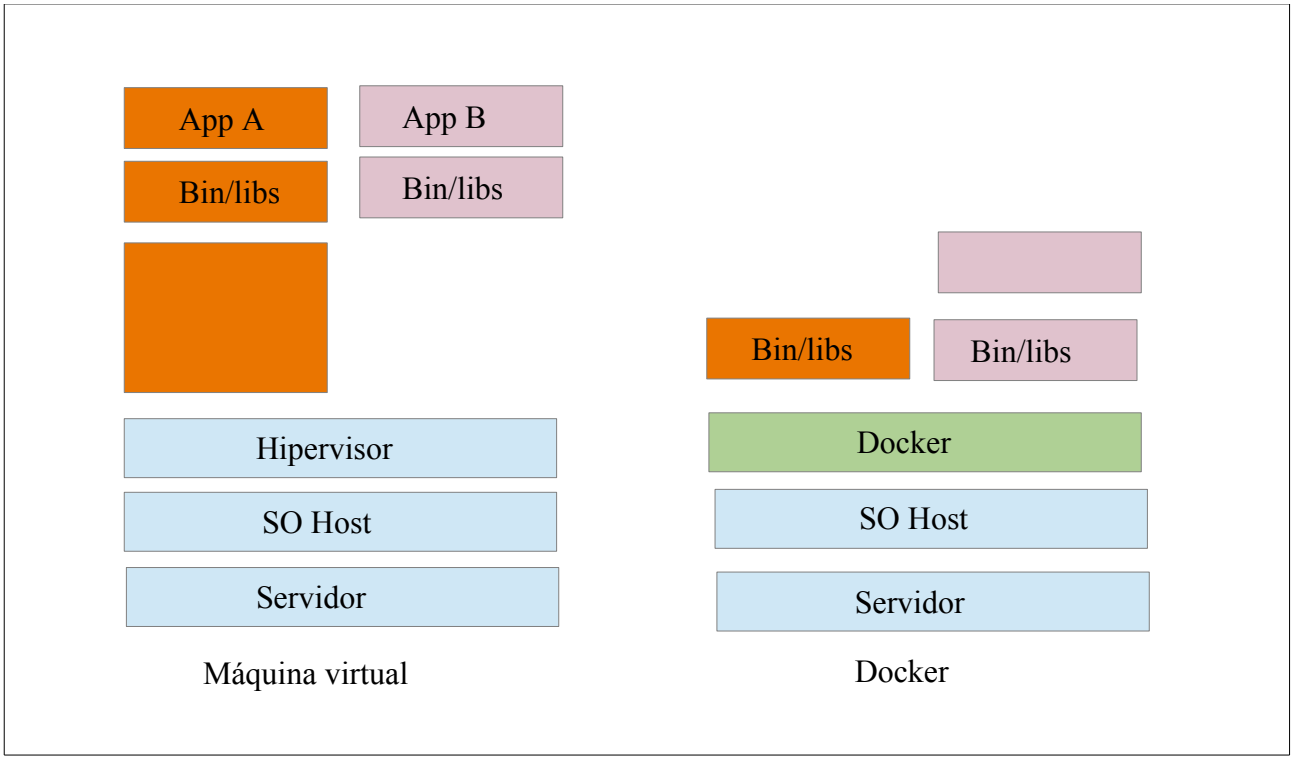
Centro de Procesamiento de Datos

Práctica 3. Contenedores Docker

Objetivo:
Introducción a contenedores Docker.

¿Qué es docker?

Docker es una plataforma que permite desarrollar, desplegar y ejecutar aplicaciones en contenedores. Los contenedores tienen múltiples ventajas ya que son:
- Ligeros: Son procesos que comparten el Kernel del sistema operativo.
- Portables: Se pueden crear localmente y ejecutar en cualquier entorno o en computación en la un nube. Además, al desplegarse en proveedores en la nube, sin tener que definir una infraestructura (máquina virtual), se reducen los costes a dichos proveedores y estos a su vez ofrecen menor coste en la ejecución.
- Intercambiables: Se pueden desarrollar nuevas actualizaciones que son fácilmente actualizables.
- Escalables: Se pueden añadir y distribuir copias.
- Flexibles: Gran cantidad de aplicaciones pueden ser incluidas en un contenedor.



Desarrollo:
Esta práctica es una introducción a Docker para familiarizarse con los conceptos, modo de funcionamiento y órdenes básicas.

I) Instalación de Docker

Podemos utilizar Docker en diferentes sistemas operativos. En el caso de Ubuntu podemos instalarlo con apt:

```
apt install docker.io -y
```

Docker funciona como contenedores nativos Linux, es decir, son procesos que se ejecutan en el servidor como procesos del sistema operativo, compartiendo el kernel del Linux.

También podemos instalarlo en Windows y Mac. En este caso, Docker crea una máquina virtual ligera Linux que pueden ejecutar dichos contenedores.

Una vez instalado podemos ejecutar para comprobar el estado general:

```
docker info
```

II) Lista de contenedores

En https://hub.docker.com/ podemos consultar una lista de miles de contenedores preconfigurados y listos para instalarse.

Para buscar:

```
docker search  <nombre>
```

Ej: busquemos contenedores *nginx*

```
docker search nginx
```

Ej: buscar contenedores Ubuntu

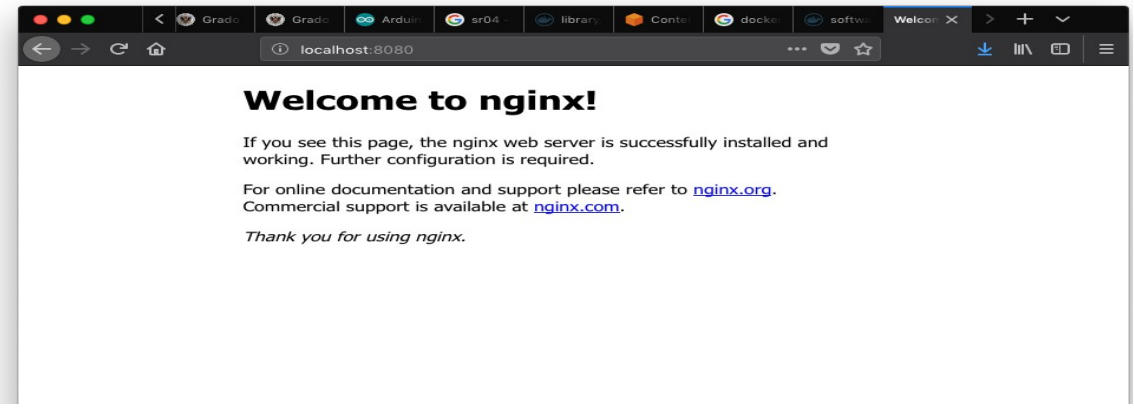
III) Instalación de un contenedor de NGINX.

Instalemos un contenedor basado en NGINX:

```
docker run --name nginx1 -p8080:80 -p8443:443 -d nginx
```

A partir de la imagen oficial de nginx creamos el contenedor nginx1, que expone los puertos 80 y 443 del contenedor en los puertos 8080 y 8443 del host. Además el contenedor se inicia tras la ejecución.

Comprobamos que podemos acceder al servidor NGINX en *http://localhost:8080*



También se podría utilizar la opción -P para exponer los puertos 80 y 443 de forma dinámica, correspondiendo con puertos libres del *host*.

La orden ps nos muestra los contenedores que están ejecutándose:

```
docker ps
```

Para ver todos los contenedores:

```
docker ps -a
```

Parar un contenedor:

```
docker stop <id_contenedor>
```

Iniciar un contenedor:

```
docker start <id_contenedor>
```

Borrar un contenedor:

```
docker rm <id_contenedor>
```

Información detallada de un contenedor :

```
docker inspect <id_contenedor>
```

Estadística de funcionamiento: %CPU, uso de memoria, E/S, …

```
docker stats
```

III) Imágenes Docker

Docker almacena las imágenes en una caché, lo que permite crear nuevos contenedores en un tiempo reducido y aprovechando recursos de almacenamiento.

Para ver las imágenes almacenadas:

```
docker images
```

Para borrar una imagen:

```
docker rmi <id_imagen>
```

Para descargar una imagen:

```
docker pull  <nombre_imagen>
```

Más adelante veremos como es posible utilizar un contenedor, añadir nuevos paquetes y personalizarlo para crear nuevos contenedores.

IV) Compartiendo el directorio host con el contenedor

Docker separa su almacenamiento en dos: Por una parte las aplicaciones quedan almacenadas en imágenes fijas (imagen de un contenedor), y por otra parte el almacenamiento de datos.

Podemos montar

```
docker run --name nginx2 -v /var/www:/usr/share/nginx/html:ro -v /var/nginx/conf:/etc/nginx:ro -p8081:80 -p8444:443 -d nginx
```

Comparte los directorios /var/www y /var/nginx/conf del host con los directorios /usr/share/nginx/html y /etc/nginx del contenedor.

Para que nginx pueda releer el fichero de configuración:

```
docker kill -s HUP nginx2
```

Esto envía una señal al proceso.

Otro ejemplo con busybox:

```
docker run -it --name container1 -v /home/usuario/datos_c1:/datavol busybox
```

IV) Crear volúmenes para almacenamiento

```
docker run -it -v /data --name container1 busybox
```

Podemos acceder al directorio /data en el contenedor y crear algún ficheros

Salimos del contenedor

```
exit
```

Comprobamos

```
docker ps -a
```

Consultar

```
docker inspect container1
```

V) Ejecución interactiva

Podemos crear un contenedor

```
docker run -it ubuntu bash
```

Nos salimos con exit o con Control-D.

Si queremos renombrar un contenedor:

```
docker rename <id_contenedor> nombre
```

Reniciamos y accedemos

```
docker restart mi_contenedor
docker attach mi_contenedor
```

VI) Creación de contenedor de forma interactiva

Aunque podemos encontrar bastantes contenedores preconfigurados,
https://hub.docker.com/explore/
es útil poder definir nuestro propio contenedor.

Crear de forma interactiva

```
docker run -i -t ubuntu bash
```

Dentro del contenedor:
Actualizamos

```
apt update
apt upgrade
```

Instalamos nginx:

```
apt install nginx
```

Salimos del contenedor con exit

Guardamos la imagen:

```
docker ps -a
docker commit <id_contenedor> mi_imagen
```

Comprobamos que la imagen está disponible:

```
docker images
```

Una vez creada una imagen del contenedor, podemos compartirla con un compañero.
Para ello, se guarda la imagen en formato .tar mediante:

```
docker save -o <ruta del fichero> <nombre imagen>
```

En el otro nodo se recupera la imagen mediante:

```
docker load -i <ruta del fichero>
```