Actividad 5 (v. 200701) Pablo González Nalda

Depto. de *Lenguajes y Sistemas Informáticos* lsi.vc.ehu.eus/pablogn

Universidad del País Vasco Euskal Herriko Unibertsitatea

GASTEIZKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE VITORIA-GASTEIZ

Actividad 5 Vagrant y Docker

Métodos de virtualización, en máquina virtual y en contenedores. Concepto de *microservicio*.

Vagrant

Automatización de máquinas virtuales y contenedores, con diferentes hipervisores o *proveedores*: VirtualBox, Hyper-V, VMware, Docker...

Se usa un fichero `Vagrantfile` que describe la máquina virtual, por lo que no hace falta un proceso de asistente de creación o *wizard*. Se pueden crear varias máquinas en un solo fichero. Se descarga una plantilla o `box` con una instalación de la máquina que posteriormente se *provisiona* como se indica en el `Vagrantfile`. Provisionar consiste en que se personaliza y parametriza la máquina, incluyendo la operación de instalar software y prepararlo para que desarrolle una cierta tarea.

Más información: https://www.vagrantup.com/intro/index.html

Getting started: https://www.vagrantup.com/intro/getting-started/index.html

A list of boxes: http://www.vagrantbox.es/

Search: https://app.vagrantup.com/boxes/search

Uso:

```
vagrant box update
vagrant up
vagrant ssh
vagrant provision
vagrant destroy -f
```

Docker

En esta práctica daremos los primeros pasos en Docker y observaremos sus principales características.

2.1. Instalación

Como indica en https://docs.docker.com/engine/install/ para la instalación hay muchas variantes.

En Windows https://docs.docker.com/docker-for-windows/install/

En Ubuntu:

```
sudo apt install docker.io
sudo groupadd docker
sudo usermod -aG docker $USER
```

Es necesario como mínimo salir de la sesión y volver a entrar para que la cuenta usuaria actualice los grupos. Se comprueban con la instrucción `id`.

En las salas de ordenadores de la escuela ya está instalado, en la cuenta `udocker`.

2.2. Manejo básico

Para arrancar un microservicio (en este caso sólo es un comando de ubuntu para comprobar que se usa el mismo kérnel y borra con `-rm` el contenedor):

```
docker pull ubuntu
docker images # imágenes, tenemos el ubuntu, observad el tamaño
docker run --rm ubuntu uname -a # ejecuta el comando

docker run --rm -it ubuntu bash # interactivo, línea de comandos
  sleep 100 & # en segundo plano
  ps aux # vemos su PID, el del bash que arranca el contenedor y el ps aux
# desde fuera del contenedor, otra línea de comandos
ps aux | grep sleep # vemos que es el único sleep con otro PID

docker ps -a # contenedores en ejecución y parados con el -a, se han borrado
docker run ubuntu uname -a # no borra el contenedor
docker ps -a # contenedores en ejecución y parados con el -a
docker rm practical_lehmann # borra el contenedor usando el nombre de NAMES
docker images # sigue estando la imagen
docker rmi ubuntu # borra la imagen
```

Fichero `Dockerfile` mínimo para crear una imagen:

```
FROM scratch
COPY busybox /
```

Fichero 2.1: `Dockerfile`

Crear la imagen con

```
cp /bin/busybox .
docker build --tag ejemplo .
```

Arrancar para hacer un `ls -l` dentro del contenedor con

```
docker run --rm ejemplo ./busybox ls -l
```

y para crear una línea de comandos interna:

```
docker run -it --rm --name "EjemploDocker" -h "ED" ejemplo ./busybox ash
# y dentro por ejemplo:
uname -a
exit  # para salir
docker run -it --rm --name "Ejdocker" -h "ED" ejemplo ./busybox pwd
/ # devuelve pwd que es la raíz de su sistema de ficheros del contenedor
```

2.3. Servidor web

Para arrancar un microservicio con un solo servicio servidor web apache. Lo arranca como demonio (en segundo plano), con nombre *web* y con puerto externo 8080 e interno 80, imagen httpd y un volumen *sitio* para ese directorio que contiene la web.

```
docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/ httpd
docker ps
docker volume ls
docker volume inspect sitio
sudo ls -l /var/lib/docker/volumes/sitio/_data
sudo vi /var/lib/docker/volumes/sitio/_data/index.html
```

Para ver que funciona accedemos http://localhost:8080/ y para comprobar que la modificación funciona.

También podemos entrar dentro del contenedor para editar.

```
docker ps -a
docker start web # si ha parado
docker exec -it web bash # un nuevo proceso dentro del contenedor
apt update
apt install vim
cd htdocs
vim index.html
```

Y podemos ver el cambio en http://localhost:8080/. Luego paramos y borramos el contenedor, y cuando arrancamos otro se siguen viendo los cambios.

```
docker stop web
docker rm web
docker ps -a
docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/ httpd
```

2.4. docker-compose

Hay tres formas de instalar: dos son bajando el fichero y otra por **apt**. Instalación de `docker-compose` en `/usr/local/bin` como **sysadmin** (con `sudo`) o en un fichero **local** por si no tenemos permisos de `root` como en las aulas:

```
sudo curl -L  "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)
    -$(uname -m)" -o /usr/local/bin/docker-compose
curl -L  "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(
    uname -m)" -o docker-compose # fichero local, ejecutar con ./docker-compose
```

Para unificar, usamos un *alias*: `alias docker-compose='./docker-compose'`

Por `apt`

```
sudo apt install docker-compose
```

Para arrancar el grupo de contenedores Docker especificados en el `docker-compose.yml` que levanta el servicio Nextcloud se hace:

```
cd nextcloud/
docker-compose up -d
docker ps -a
docker-compose logs # evolución, registros
```

En `http://localhost:88/` se abre la configuración. Hay que poner nombre y contraseña (nuevos) y los datos del YML: base de datos MariaDB, usuario *nextcloud*, *Contraseña*, nombre de la BD *nextcloud* y host de la BD *db:3306* (es el puerto por defecto de MariaDB).

Para parar:

```
docker-compose down # borra contenedores pero no datos
docker volume ls    # listado de volúmenes
docker volume prune # borra los volúmenes no usados
```

Si no borramos los volúmenes podemos volver a arrancar con los mismos datos y con nuevos contenedores, actualizando versiones si se especifica la etiqueta o *tag* `latest` u otra en el YML.

A continuación, el resultado de `history` después de hacer el laboratorio:

```
1875  ls -l
1876  docker pull ubuntu
1877  docker images
1878  docker run --rm ubuntu uname -a
1879  uname -a
1880  docker run --rm it ubuntu bash
1881  ps aux|grep sleep
1882  docker run --rm -it ubuntu bash
1883  docker ps -a
1884  docker run ubuntu ps aux
1885  docker ps -a
1886  docker rm gifted_edison
1887  docker images
1888  docker rmi ubuntu emilevauge/whoami
1889  docker images
1890  ls -l
1891  cd busybox/
1892  l
1893  cat Dockerfile
1894  cp /bin/busybox .
1895  docker build --tag ejemplo .
```

```
1896  docker images
1897  docker run --rm ejemplo ./busybox ls -l
1898  docker run --rm ejemplo ./busybox ps aux
1899  docker inspect ejemplo
1902  sudo ls -l /var/lib/docker/overlay2/d7328064975c064975c23b5379d982c2857c9bdbbb2e0/diff
1903  docker run -it --rm --name "Ejdocker" -h "ED" ejemplo ./busybox ash
1904  docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/ httpd
1905  curl localhost:8080
1906  docker ps -a
1907  docker volume ls
1908  docker volume inspect sitio
1909  sudo ls -l /var/lib/docker/volumes/sitio/_data
1910  sudo vi /var/lib/docker/volumes/sitio/_data/index.html
1911  curl localhost:8080
1912  docker stop web
1913  docker ps -a
1914  docker rm web
1915  docker ps -a
1916  docker run -dit --name web -p 8080:80 -v sitio:/usr/local/apache2/htdocs/ httpd
1917  curl localhost:8080
1918  cd ..
1919  docker stop web
1920  docker rm web
1921  ls
1922  cd nextcloud
1923  ls
1924  cat docker-compose.yml
1925  ls
1926  docker-compose up -d
1927  curl localhost:88
1928  docker-compose ps
1929  docker ps -a
1930  docker-compose logs
1932  docker-compose logs app
1933  docker-compose down
1934  docker volume ls
1935  docker volume prune
1936  cd ..
```

Fichero 2.2: `history`