docker, contenedores

Docker

Para entendernos, esta herramienta nos permite desplegar imágenes de SO, (para nosotros son casi como máquinas virtuales) en lo que se denomina contenedores de software. El contenedor en si es una virtualización a nivel de sistema operativo con una capa de abstracción y automatización en la que está incluida todo lo necesario para su funcionamiento.

Al incluir el contenedor un filesystem completo que contiene todo lo necesario para ejecutar una aplicación (librerías, fuentes, etc) garantizamos que el software siempre correrá igual independiente del entorno en que se ejecute.

La principal diferencia con una máquina virtual es que sólo contiene lo mínimo necesario para ejecutar la aplicación, sin necesidad de un S.O completo. Además de que se ejecuta en un proceso aislado, en el espacio de usuario del S.O del host, compartiendo el kernel con otros contenedores.

La filosofía de Docker es virtualizar lo mínimo necesario para ejecutar una aplicación, sin la necesidad de tener un sistema operativo completo como host, sino que este puede ser compartido por más contenedores con el consiguiente ahorro de recursos.

Las ventajas de usar docker serían:

- Tener un mismo entorno para producción, testeo y desarrollo
- Escalado Horizontal
- Despliegue de aplicaciones con versiones antiguas.

Existe un repositorio de imágenes para Docker que se llama Registry Hub https://registry.hub.docker.com/ y desde cualquier lugar podemos, crear, compartir y consumir imágenes creadas por nosotros o por terceros.

Comparativa de Docker vs máquinas virtuales → http://www.rediris.es/tecniris/archie/doc/TECNIRIS47-3b.pdf

Conceptos

Docker basa su uso en:

- Cgroups para restringir recursos
- Kernel namespace para aislar y virtualizar recursos
- Filesystem de unión (https://en.wikipedia.org/wiki/UnionFS)

A la hora de trabajar con Docker hay que tener en cuenta los siguientes conceptos:

- imagen → es un conjunto de aplicaciones/máquina virtual empaquetada en un fichero con todo lo necesario (dependencias, configuración, etc…). No cambia y no tiene estados
- Dockerfile→ es un archivo donde definimos las reglas para crear una imagen
- contenedor → es el resultado de ejecutar una imágen(instancia), se podría decir que un contenedor es como una máquina virtal ligera., aunque en realidad es un **proceso** totalmente asilado del resto de procesos de la máquina sobre la que se ejecuta. Sus principales

características son : su portabilidad, inmutabilidad y ligereza

Source

Cuando tenemos varios entornos de docker y queremos gestionarlos desde una misma máquina podemos definir ficheros con las variables de entorno y usar el comando source para cargarlas con el comando

```
source /path/mificherovariables.sh
```

Un ejemplo de ficheros con variables sería :

```
export DOCKER_TLS_VERIFY=1
export DOCKER_CERT_PATH=/home/users/lc/certificados
export DOCKER_HOST=tcp://miservidor.aws.dckr.io:443
```

Proxy

Para que docker haga uso de un proxy debemos de crear/modificar el fichero dentro de la carpeta home del usuario desde donde vamos a lanzar docker **~/.docker/config.json** y añadir las siguientes líneas

```
{
 "proxies":
 {
   "default":
   {
     "httpProxy": "http://127.0.0.1:3001",
     "httpsProxy": "http://127.0.0.1:3001",
     "noProxy": "*.test.example.com,.example2.com"
   }
 }
}
```

Otra opción sería usar variables de entorno al llamar a docker, por ejemplo

```
docker run --env HTTP_PROXY="http://127.0.0.1:3001"
```

Proxy en Centos 7

Para que docker usara el proxy con Centos 7 Creamos la arpeta para configurar el servicio de docker a través de systemd.

```
 mkdir /etc/systemd/system/docker.service.d
```

Creamos el fichero de configuración del servicio

```
vi /etc/systemd/system/docker.service.d/http-proxy.conf
```

Añadimos a dicho fichero

```
[Service]
Environment="HTTP_PROXY=http://miproxy:8080/"
"NO_PROXY=localhost,127.0.0.0/8,10.0.0.0/8,192.168.0.0/16,172.16.0.0/12"
```

Recargamos systemctl para que tome los cambios

```
sudo systemctl daemon-reload
```

Verificamos si el entorno del servicio de docker carga correctamente

```
sudo systemctl show docker --property Environment
```

Reiniciamos el servicio

```
sudo systemctl restart docker
```

- https://docs.docker.com/config/daemon/systemd/

Recomendaciones de seguridad

Bastionado

En entornos de producción podemos ejecutar el script docker-bench-security para comprobar que cumplimos ciertos requisitos de seguridad.

- https://benchmarks.cisecurity.org/tools2/docker/CIS_Docker_1.6_Benchmark_v1.0.0.pdf
- http://www.securitybydefault.com/2015/05/buenas-practicas-de-seguridad-en-docker.html

Distribuciones con Docker

- https://vmware.github.io/photon/ Entorno optimizado para correr en servidores vmware

Monitorización de contenedores

- https://github.com/google/cadvisor

Referencias

- https://recetas-docker.readthedocs.io/es/latest/capitulo_1.html
- https://www.docker.com/community-edition
- https://docs.docker.com/installation/ubuntulinux/#installation
- https://d3oypxn00j2a10.cloudfront.net/assets/img/Docker%20Security/WP_Intro_to_container_security_03.20.2015.pdf
- http://www.cisecurity.org/
- https://benchmarks.cisecurity.org/tools2/docker/CIS_Docker_1.6_Benchmark_v1.0.0.pdf
- http://codehero.co/como-instalar-y-usar-docker/
- http://www.joseangelfernandez.es/blog/2015/03/preguntas-frecuentes-sobre-docker-para-usuarios-de-windows/
- https://www.nessys.es/docker/
- http://picodotdev.github.io/blog-bitix/2014/11/inicio-basico-de-docker/
- https://platzi.com/blog/desplegar-contenedores-docker/
- https://platzi.com/blog/imagenes-con-dockerfile/
- https://platzi.com/blog/multiples-contedenores-docker/
- https://www.gitbook.com/book/jsitech1/meet-docker/details
- https://iesgn.github.io/curso_docker_2021/
- https://fp.josedomingo.org/iawgs/u06/