DevOps Sesión 2 (2022-02-14) Docker

Documentación relacionada

- /etc/docker/daemon.json

```
 {
"bip": "172.17.0.1/16",
"ip": "192.168.33.10",
"hosts": [
        "unix:///var/run/docker.sock",
        "tcp://0.0.0.0:2376"
    ]
}
```

- Libros:
  - terraform curso practico formación Ed. RC → https://www.llardelllibre.cat/cat/libro/terraform-curso-practico-de-formacion_995340
  - rclibros.es
  - «Seguridad Docker» → https://rclibros.es/?s=docker&post_type=product&dgwt_wcas=1
  - packt → https://www.packtpub.com/
- Utilización de un Dockerfile.pdf

Clase

- exportar contenedor: `docker export`
- importar contenedor (como imagen): `docker import`
- guardar imagen: `docker save`
- cargar imagen: `docker load`
- guardar contenedor en marcha como imagen: `docker commit`
- **k0sctl**: instalación kubernetes?
- política arranque contenedores: **–restart <politica>**
  - off
  - on-failure
  - always
  - unless-stopped
  - never
- listado de procesos de un contenedor; `docker top <container>`
- docker system
  - df
  - events
  - info
  - prune
- repositorio local imagenes docker
  - https://goharbor.io (VMWARE)
  - nexus
  - artefactory
- ubicación credenciales (sólo docker-hub?): **$HOME/.docker/config.json**
- «Laboratorio docker registry.txt»

- Dockerfile
  - FROM scratch
  - FROM debian-slim
  - HEALTHCHECK -interval=2m -timeout=3s -retries=3 CMD curl -f http://127.0.0.1:5000/ || exit 1
  - USER (add-user.sh)
  - CMD VS ENTRYPOINT:

```
##CMD y ENTRYPOINT##
El ENTRYPOINT especifica un comando que siempre se ejecutará cuando
se inicie el contenedor.

El CMD especifica los argumentos que se ENTRYPOINT al ENTRYPOINT .

Si desea hacer una imagen dedicada a un comando específico,
utilizará ENTRYPOINT ["/path/dedicated_command"]

De lo contrario, si desea hacer una imagen para fines generales,
puede dejar ENTRYPOINT especificar y usar CMD
["/path/dedicated_command"] ya que podrá anular la configuración al
proporcionar argumentos a la docker run .

#Por ejemplo, si su Dockerfile es:

FROM debian:wheezy
ENTRYPOINT ["/bin/ping"]
CMD ["localhost"]

#Ejecutar la imagen sin ningún argumento hará ping al localhost:

$ docker run -it test
PING localhost (127.0.0.1): 48 data bytes
56 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.096 ms
56 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.088 ms
56 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.088 ms
^C--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.088/0.091/0.096/0.000 ms

#Ahora, ejecutar la imagen con un argumento hará ping al argumento:

$ docker run -it test google.com
PING google.com (173.194.45.70): 48 data bytes
56 bytes from 173.194.45.70: icmp_seq=0 ttl=55 time=32.583 ms
56 bytes from 173.194.45.70: icmp_seq=2 ttl=55 time=30.327 ms
56 bytes from 173.194.45.70: icmp_seq=4 ttl=55 time=46.379 ms
^C--- google.com ping statistics ---
5 packets transmitted, 3 packets received, 40% packet loss
round-trip min/avg/max/stddev = 30.327/36.430/46.379/7.095 ms

##Para comparación, si su Dockerfile es:

FROM debian:wheezy
CMD ["/bin/ping", "localhost"]

#Ejecutar la imagen sin ningún argumento hará ping al localhost:
```

```
$ docker run -it test
PING localhost (127.0.0.1): 48 data bytes
56 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.076 ms
56 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.087 ms
56 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.090 ms
^C--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.076/0.084/0.090/0.000 ms

#Pero ejecutar la imagen con un argumento ejecutará el argumento:

docker run -it test bash
root@e8bb7249b843:/#

docker run -it test echo hola

##En nuestra imagen de jboss ENTRYPOINT
$JBOSS_HOME/bin/standalone.sh
##Lanzamos la imagen arrancando solo el proceso de bash en el
contenedor
docker run -dit  --entrypoint bash  --name jboss1  jboss-eap-mysql-
centos-jdk
```

- `dockerfile build –no-cache –rm –pull -t <IMAGEN>:<TAG> .`
  - **–no-cache**: no usar cache
  - **–rm** eliminia contenedores intermedios del **build**
  - **–pull**: descargará los FROM de nuevo en el momento del build (para no reaprovechar las imágenes viejas descargadas)

Extras

- gosu: https://github.com/tianon/gosu
  - mysql Dockerfile: https://github.com/docker-library/mysql/blob/aa600026fe54b1fa6b2a7ac80ffbb466618fcabf/8.0/Dockerfile.debian

From:
https://matebcn.eu/ - **miguel angel torres egea**

Permanent link:
https://matebcn.eu/info:cursos:pue:devops2022:s2

Last update: **09/03/2022 18:57**