

# DevOps Sesión 1 (2022-02-09) Docker

## Documentación relacionada

- montar entorno docker con Vagrant:
  - Entorno docker en PUE.txt
    - **git clone** <https://github.com/agarciafer/lab-docker.git>
  - Entorno de mv de docker-local.txt

## 1-Despliegue de Aplicaciones Docker

- Seminario Herramientas Devops entornos de desarrollo en local, para programadores con Docker y Vagrant .pdf
- 1-Curso Alumnos Docker.pdf
  - pag 55
  - pag 66
    - Clase1-docker.txt
    - <https://docs.docker.com/engine/reference/commandline/>
    - **/var/lib/docker**
    - `docker search <busqueda>` → <https://hub.docker.com>
    - `docker search -f is-official=true nginx`
    - `docker inspect <IMAGEN>`
    - `docker logs <CONTAINER>`
  - pag 88 - 94 : laboratorio 3
    - `docker attach` → <https://docs.docker.com/engine/reference/commandline/attach/>
    - `docker stats`
    - `docker update` → establecer límites contenedor en ejecución
- navegar entre capas imagen docker: <https://github.com/wagoodman/dive>

## extras

```
##Personalización del servidor de Docker
https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file

##Paramos docker antes de realizar el procedimiento:
systemctl stop docker

systemctl cat docker.service
[root@docker1 docker]# export EDITOR=vi
[root@docker1 docker]# systemctl edit --full docker.service

#ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

systemctl daemon-reload
systemctl restart docker
systemctl status docker
```

### ##Personalizar el daemon de docker a traves de /etc/docker/daemon.json:

```
cat /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"
  ],
  "insecure-registries": [
    "docker1.curso.local:5000"
  ],
  "tlscacert": "/ca/ca.pem",
  "tlscert": "/ca/server-cert.pem",
  "tlskey": "/ca/server-key.pem",
  "tlsverify": true
}
```

```
sytemctl daemon-reload
systemctl restart docker
systemctl status docker
```

```
/etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment="HTTPS_PROXY=https://proxy.example.com:443/"
"NO_PROXY=localhost,127.0.0.1,docker-registry.example.com,.corp"
```

```
sytemctl daemon-reload
systemctl restart docker
systemctl status docker
```

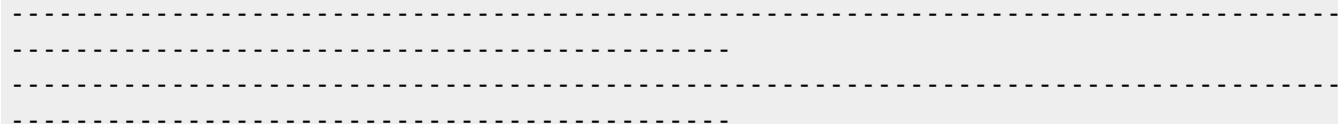
```
[root@docker1 docker]# export EDITOR=vi
[root@docker1 docker]# systemctl edit --full docker.service
```

```
[Service]
Environment=http_proxy=http://10.28.24.99:3128/
```

```
sytemctl daemon-reload
systemctl restart docker
systemctl status docker
```

El archivo de configuración del demonio daemon.json, se ubicará de forma predeterminada en las siguientes ubicaciones:

- /etc/docker/daemon.json en sistemas Linux
- %programdata%\docker\config\daemon.json en sistemas Windows



### ##Personalización del cliente de Docker

El cliente almacenará su configuración en el directorio de inicio de los usuarios en `$HOME/.docker`  
Hay un archivo de configuración donde el cliente Docker buscará sus configuraciones (`$HOME/.docker/config.json` en Linux o `%USERPROFILE%/.docker/config.json` en Windows)

Si necesitamos pasar la configuración del proxy a los contenedores al inicio, configuraremos la proxies clave `.docker/config.json` para nuestro usuario, por ejemplo,

usando `my-company-proxy`:

```
"proxies":
{
"default":
{
"httpProxy": "http://my-company-proxy:3001",
"httpsProxy": "https://my-company-proxy:3001",
"noProxy": "/*.test.example.com,.example2.com"
}
}
```

## Estas configuraciones se pueden agregar como argumentos al iniciar el contenedor Docker, de la siguiente manera `docker run`:

```
--env HTTP_PROXY="http://my-company-proxy:3001"
--env HTTPS_PROXY="https://my-company-proxy:3001"
--env NO_PROXY="/*.test.example.com,.example2.com"
```

#####LABORATORIO Seguridad docker tls daemon#####:  
##Procedimiento pagina 45 pdf Tema2:

Editar en `docker1` y `docker2`:

```
vi /etc/hosts
192.168.33.10 docker1.curso.local docker1
192.168.33.11 docker2.curso.local docker2
```

##Todo en la `mv docker1`:

```
mkdir /ca
cd /ca
```

##Declaramos las variable local:

```
export HOST=docker1.curso.local
echo $HOST
```

```
openssl genrsa -aes256 -out ca-key.pem 4096
openssl req -new -x509 -days 365 -key ca-key.pem -sha256 -out ca.pem
```

Common Name (e.g. server FQDN or YOUR name) []:docker1.curso.local

```
openssl genrsa -out server-key.pem 4096
openssl req -subj "/CN=$HOST" -sha256 -new -key server-key.pem -out server.csr
echo subjectAltName = DNS:$HOST,IP:192.168.33.10,IP:127.0.0.1 >> extfile.cnf
echo extendedKeyUsage = serverAuth >> extfile.cnf
```

```
openssl x509 -req -days 365 -sha256 -in server.csr -CA ca.pem -CAkey ca-key.pem -
CAcreateserial -out server-cert.pem -extfile extfile.cnf
```

```
$ chmod -v 0400 ca-key.pem key.pem server-key.pem
$ chmod -v 0444 ca.pem server-cert.pem cert.pem
```

```
vi /etc/docker/daemon.json
{
  "bip": "172.17.0.1/16",
  "hosts": [
    "unix:///var/run/docker.sock",
    "tcp://0.0.0.0:2376"
  ],
  "insecure-registries": [
    "docker1.curso.local:5000"
  ],
  "tlscacert": "/ca/ca.pem",
  "tlscert": "/ca/server-cert.pem",
  "tlskey": "/ca/server-key.pem",
  "tlsverify": true
}
```

###Editamos el servicio de docker y comentamos la linea ExecStart ahora añadimos la linea ExecStart=/usr/bin/dockerd:

```
export EDITOR=vi
systemctl edit --full docker.service
```

```
# /etc/systemd/system/docker.service
#ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
ExecStart=/usr/bin/dockerd
```

###Ahora podemos arrancar nuestro docker engine en el servidor docker1:

```
systemctl daemon-reload
systemctl restart docker
systemctl status docker
```

```
netstat -tan |grep -i 2376
tcp6      0      0 :::2376          :::*              LISTEN
```

-----  
###En la mv docker1 creamos los certificados para cliente:

#Con la misma CA, cree una clave firmada por la CA del cliente, especificando que esta clave se utilizará para la autenticación del cliente: \$ openssl genrsa -out key.pem 4096

```
cd /ca
openssl genrsa -out key.pem 4096
openssl req -subj '/CN=docker2.curso.local' -new -key key.pem -out client.csr
echo extendedKeyUsage = clientAuth > extfile-client.cnf
openssl x509 -req -days 365 -sha256 -in client.csr -CA ca.pem -CAkey ca-key.pem -
```

```
Ccreateserial -out cert.pem -extfile extfile-client.cnf

##En la mv docker2:
mkdir /ca-cliente

##Moveremos los certificados de cliente generados al host del cliente
Tendremos que copia desde la mv docker1 en /ca al directorio de la mv docker2 /ca-
cliente:
ca.pem
cert.pem
key.pem

##Desde docker1:
cd /ca
scp ca.pem 192.168.33.11:/ca-cliente
scp cert.pem 192.168.33.11:/ca-cliente
scp key.pem 192.168.33.11:/ca-cliente

##En la mv docker2 en el directorio /ca-cliente, tendremos que tener los
siguientes archivos copiados de la mv docker1:

cd /ca-cliente
ls
ca.pem
cert.pem
key.pem

##Ahora podemos comprobar el funcionamineto desde el cliente docker2 en el
directorio /ca-cliente donde estan los certs del cliente:
##Con este comando nos tiene que fallar:

docker -H=docker1.curso.local:2376 version

[root@docker2 ca-cliente]# docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem
--tlskey=key.pem -H=docker1.curso.local:2376 version
[root@docker2 ca-cliente]# docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem
--tlskey=key.pem -H=docker1.curso.local:2376 ps -a
[root@docker2 ca-cliente]# docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem
--tlskey=key.pem -H=docker1.curso.local:2376 info

-----
-----
https://docs.docker.com/engine/security/protect-access/

##No es necesrio para el laboratorio:

##Personalizar el cliente docker2:

vi /root/.docker/config.json
{
```

```
"auths": {},
}

##Por defecto en el cliente busca los certificados:
/root/.docker/
ca.pem
cert.pem
key.pem

docker -H=docker1.curso.local:2376 info

##Este comportamiento lo podemos cambiar:

export DOCKER_CERT_PATH=/ca-cliente
export DOCKER_TLS_VERIFY="1"

docker -H=docker1.curso.local:2376 info

###Para dejarlo fijo para el usuario root:
vi /root/.bash_profile
export DOCKER_CERT_PATH=/ca-cliente
export DOCKER_TLS_VERIFY="1"

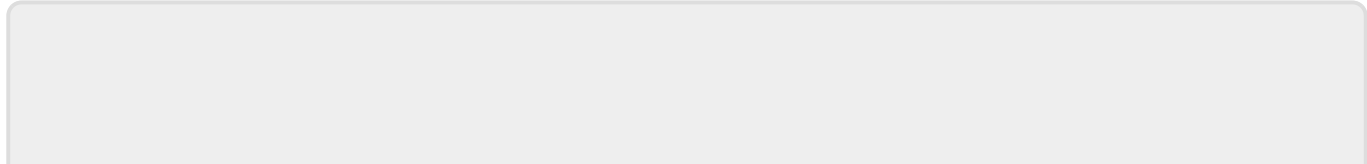
-----
-----
-----

##Para modo depuracion sin utilizar systemd:

Inicie el demonio de Docker con TLS habilitado y use argumentos para la CA, el
certificado del servidor y la clave firmada por la CA.
Esta vez, el demonio de Docker que usa TLS se ejecutará en el puerto 2376 (que es
estándar para el demonio TLS):

# which dockerd
/usr/bin/dockerd

##Para poder ejecutar el debugger del demonio dockerd, tendremos que tener renombrado
el archivo /etc/docker/daemon.json
cd /ca
dockerd --tlsverify --tlscacert=ca.pem --tlscert=server-cert.pem --tlskey=server-
key.pem -H=0.0.0.0:2376
```



From:

<https://miguelangel.torresegea.es/wiki/> - **miguel angel torres egea**

Permanent link:

<https://miguelangel.torresegea.es/wiki/info:curros:pue:devops2022:s1>

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

