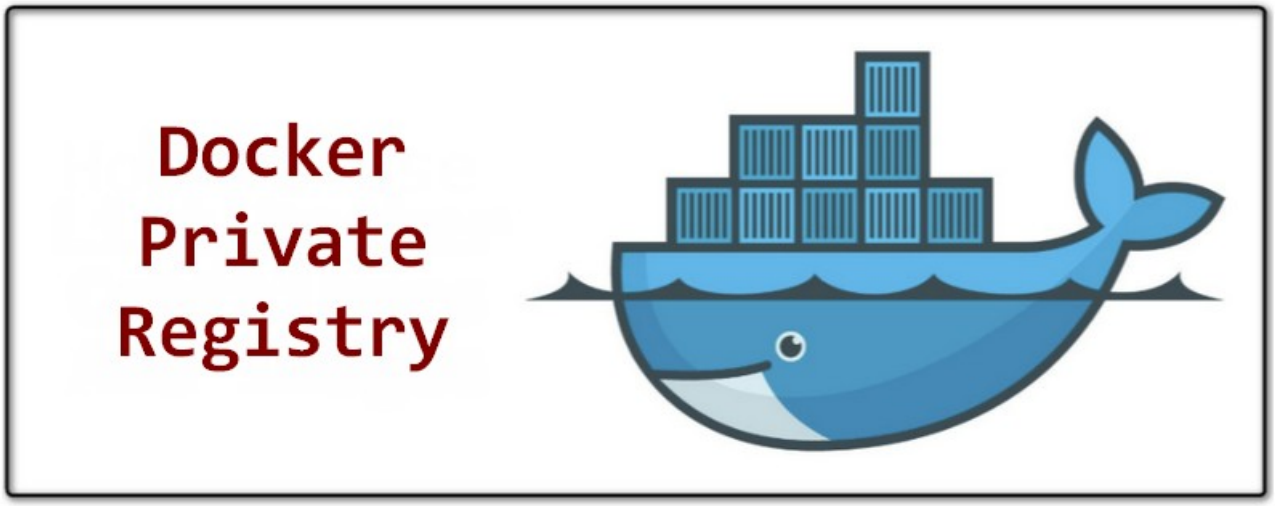

Docker Registry

IES Gonzalo Nazareno

Juan Diego Abadía Noguera

Proyecto Fin de Ciclo

01/12/20

Sumario

1. Introducción

En este proyecto vamos a montar un docker registry (es un sistema de almacenamiento y distribución para imágenes de Docker) con un proxy nginx, autentificación básica y https

2. Docker Registry

1. Entidad certificadora

- Creamos el entorno de la unidad certificadora

```
vagrant@buster:~$ sudo su -
root@buster:~# mkdir ca
root@buster:~# DIR_CA="/root/ca"
root@buster:~# cd $DIR_CA
root@buster:~/ca# mkdir certs csr crl newcerts private
root@buster:~/ca# chmod 700 private
root@buster:~/ca# touch index.txt
root@buster:~/ca# touch index.txt.attr
root@buster:~/ca# echo 1000 > serial
```

- Especificamos las variable necesarias para crear los certificados

```
root@buster:~/ca# countryName_default="ES"
root@buster:~/ca# stateOrProvinceName_default="Sevilla"
root@buster:~/ca# localityName_default="Dos Hermanas"
root@buster:~/ca# organizationName_default="juandi.com"
root@buster:~/ca# organizationalUnitName_default="Juandi"
root@buster:~/ca# emailAddress_default=""
```

- Creamos el fichero openssl.conf para configurar la unidad certificadora

```
DIR_CA="./"
cat <<EOF>$DIR_CA/openssl.conf
[ ca ]
# man ca
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir               = ${DIR_CA}
certs             = ${DIR_CA}certs
crl_dir           = ${DIR_CA}crl
new_certs_dir     = ${DIR_CA}newcerts
database          = ${DIR_CA}index.txt
serial            = ${DIR_CA}serial
RANDFILE          = ${DIR_CA}private/.rand

# The root key and root certificate.
private_key       = ${DIR_CA}private/juandi.key.pem
certificate       = ${DIR_CA}certs/juandi.cert.pem

# For certificate revocation lists.
crlnumber         = ${DIR_CA}crlnumber
crl               = ${DIR_CA}crl/ca.crl.pem
crl_extensions    = crl_ext
default_crl_days  = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md        = sha256

name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
preserve          = no
policy            = policy_strict

[ policy_strict ]
```

```
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of man ca.
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the ca man page.
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
# Options for the req tool (man req).
default_bits        = 4096
distinguished_name  = req_distinguished_name
string_mask         = utf8only
# SHA-1 is deprecated, so use SHA-2 instead.
default_md          = sha256
# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca

[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
```

```
commonName                      = Common Name
emailAddress                    = Email Address

# Optionally, specify some defaults.
countryName_default             = $countryName_default
stateOrProvinceName_default     = $stateOrProvinceName_default
localityName_default            = $localityName_default
0.organizationName_default      = $organizationName_default
organizationalUnitName_default  = $organizationalUnitName_default
emailAddress_default            = $emailAddress_default

[ v3_ca ]
# Extensions for a typical CA (man x509v3_config).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (man x509v3_config).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Extensions for client certificates (man x509v3_config).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Extensions for server certificates (man x509v3_config).
```

```
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (man x509v3_config).
authorityKeyIdentifier=keyid:always

[ ocsp ]
# Extension for OCSP signing certificates (man ocsp).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
EOF
sed -i 's|xxxsubjectAltNamexxx \=|subjectAltName = ${ENV::SAN}|g' openssl.conf
```

- Creamos la clave de la unidad certificadora

```
root@buster:~/ca# openssl genrsa -aes256 -out private/juandi.key.pem 4096

Generating RSA private key, 4096 bit long modulus (2 primes)

.....................................................................................
.....................................................................................
..++++

.....................................................................+++++

e is 65537 (0x010001)

Enter pass phrase for private/juandi.key.pem:

Verifying - Enter pass phrase for private/juandi.key.pem:
```

- Creamos el certificado de la unidad certificadora

```
root@buster:~/ca# openssl req -config openssl.conf -key private/juandi.key.pem -new -x509 -days
7300 -sha256 -extensions v3_ca -out certs/juandi.cert.pem

Enter pass phrase for private/juandi.key.pem:

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [ES]:

State or Province Name [Sevilla]:

Locality Name [Dos Hermanas]:

Organization Name [juandi.com]:

Organizational Unit Name [Juandi]:

Common Name []:juandi.com

Email Address []:juandinogueraide@gmail.com
```

- Creamos la clave y solicitud de certificado de la pagina registry.juandi.com (En este caso he creado un certificado wildcard)

```
vagrant@buster:~$ openssl req -new -newkey rsa:4096 -nodes -keyout juandi.com.key -out
juandi.com.csr
Generating a RSA private key
.....................++++
.....++++
writing new private key to 'juandi.com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Sevilla
Locality Name (eg, city) []:Dos Hermanas
Organization Name (eg, company) [Internet Widgits Pty Ltd]:juandi.com
Organizational Unit Name (eg, section) []:Juandi
Common Name (e.g. server FQDN or YOUR name) []:*.juandi.com
Email Address []:juandinogueraide@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

- Firmamos la solicitud de certificado

```
root@buster:~/ca# openssl ca -config openssl.conf -extensions server_cert -days 365 -notext -md
sha256 -in certs/juandi.com.csr -out certs/juandi.com.cert

Using configuration from openssl.conf

Enter pass phrase for ./private/juandi.key.pem:

Check that the request matches the signature

Signature ok

Certificate Details:

        Serial Number: 4096 (0x1000)

        Validity

            Not Before: Jul 14 09:07:58 2020 GMT

            Not After : Jul 14 09:07:58 2021 GMT

        Subject:

            countryName               = ES

            stateOrProvinceName       = Sevilla

            organizationName          = juandi.com

            organizationalUnitName    = Juandi

            commonName                = *.juandi.com

            emailAddress              = juandinogueraide@gmail.com

        X509v3 extensions:

            X509v3 Basic Constraints:

                CA:FALSE

            Netscape Cert Type:

                SSL Server

            Netscape Comment:

                OpenSSL Generated Server Certificate

            X509v3 Subject Key Identifier:

                B6:54:54:7F:60:60:1E:8C:39:D3:34:07:9A:9A:15:20:4D:E7:C2:39

            X509v3 Authority Key Identifier:

                keyid:97:CC:2A:FD:AC:BA:B7:AF:22:93:07:18:BF:88:BF:6F:46:8B:39:1B

                DirName:/C=ES/ST=Sevilla/L=Dos
Hermanas/O=juandi.com/OU=Juandi/CN=juandi.com/emailAddress=juandinogueraide@gmail.com

                serial:60:C9:16:11:7F:96:96:67:AE:5C:2B:B6:DF:1E:4D:76:8E:D3:7E:6B


            X509v3 Key Usage: critical

                Digital Signature, Key Encipherment

            X509v3 Extended Key Usage:
```

```
                TLS Web Server Authentication
Certificate is to be certified until Jul 14 09:07:58 2021 GMT (365 days)
Sign the certificate? [y/n]:y



1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

- Cambiamos los permisos al certificados

```
root@buster:~/ca# chmod 444 certs/juandi.com.cert
```

2. Docker registry

- Este es nuestro docker-compose con el que crearemos nuestro docker registry

```
version: '3'


services:
  registry:
    restart: always
    image: registry:2
    ports:
    - "5000:5000"
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.password
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - ./auth:/auth
      - ./data:/data
      - ./certs:/certs
```

- **Ruta donde se encuentran los usuario “auth”**
- **Ruta donde se guardaran los certificados “certs”**
- **Ruta donde se guardan las imagenes “data”**

- Configuramos nginx como proxy

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        server_name registry.juandi.com;
        return 301 https://$host$request_uri;
}

server {
         listen 443 ssl default_server;
         listen [::]:443 ssl default_server;

        ssl_certificate /etc/nginx/certs/juandi.com.cert;
        ssl_certificate_key /etc/nginx/certs/juandi.com.key;

         root /var/www/html;

         # Add index.php to the list if you are using PHP
         index index.html index.htm index.nginx-debian.html;

         server_name registry.juandi.com;

         location /v2/ {
                autoindex on;
                 # First attempt to serve request as file, then
                 # as directory, then fall back to displaying a 404.
                 # try_files $uri $uri/ =404;
                 # proxy_pass http://localhost:8080;
                 # proxy_http_version 1.1;
                 # proxy_set_header Upgrade $http_upgrade;
                 # proxy_set_header Connection 'upgrade';
                 # proxy_set_header Host $host;
                 # proxy_cache_bypass $http_upgrade;
                if ($http_user_agent ~ "^(docker\/1\.(3|4|5(?!\.[0-9]-dev))|Go ).*$" ) {
                        return 404;
        }
```

```
        proxy_pass                          http://localhost:5000;
        proxy_set_header  Host              $http_host;   # required for docker client's sake
        proxy_set_header  X-Real-IP         $remote_addr; # pass on real client's IP
        proxy_set_header  X-Forwarded-For   $proxy_add_x_forwarded_for;
        proxy_set_header  X-Forwarded-Proto $scheme;
        proxy_read_timeout                  900;
        }
}
```

- Creamos con el comando htpasswd los usuarios para logearnos en el registry

```
root@docker:~/registry/registry.juandi.com/auth# htpasswd -Bc registry.password juandi97
New password:
Re-type new password:
Adding password for user juandi97
```

- Una vez creado los usuarios, ya podemos lanzar el docker-compose. Cuando el contenedor este iniciado nos conectamos a nuestro registry

```
root@docker:~# docker login https://registry.juandi.com:5000
Username: juandi97
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

3. Prueba de funcionamiento

- Accedemos a nuestro registro de docker con el usuario creado en pasos anteriores

```
root@docker:~# docker login https://registry.juandi.com:5000

Username: juandi97

Password:

WARNING! Your password will be stored unencrypted in /root/.docker/config.json.

Configure a credential helper to remove this warning. See

https://docs.docker.com/engine/reference/commandline/login/#credentials-store


Login Succeeded
```

- Subiremos una imagen que tengamos en nuestra maquina descargada, en mi caso la imagen de nginx:alpine

```
root@docker:~# docker images

nginx                                  alpine              bd53a8aa5ac9        2 months ago
22.3MB

postgres                               10-alpine           5c6e23d6af0c        2 months ago
72.9MB

docker                                 latest              809cc4dba987        2 months ago
217MB

certbot/certbot                        latest              e294c78fd17c        3 months ago
92.1MB

mariadb                                latest              b95867b52886        4 months ago
407MB

registry                               2                   2d4f4b5309b1        5 months ago
26.2MB

juandi97/tarea2                        v1                  46f0a882b2d0        7 months ago
996MB

python                                 3                   4f7cd4269fa9        7 months ago
934MB

gestion                                latest              936ff191894b        7 months ago
542MB

mariadb                                <none>              eef18f9e510d        7 months ago
357MB

debian                                 latest              378ca4b1d2fe        7 months ago
114MB
```

- Etiquetamos la imagen con su versión y la subimos a nuestro registro

```
root@docker:~# docker tag nginx:alpine registry.juandi.com:5000/nginx:alpine


root@juandi:~# docker push registry.juandi.com:5000/nginx:alpine
The push refers to repository [registry.juandi.com:5000/nginx]
c18d9d73ad92: Pushed
c2a463dbbd57: Pushed
4ccac0888f99: Pushed
553e0f8f950d: Pushed
50644c29ef5a: Mounted from postgres
alpine: digest: sha256:a3c6118edc80de4a5aaf2711b7742c25d4d2da54325bae465205cb386afa79ee size: 1360
```

3. Conclusión

Docker registry podría ser una herramienta muy aconsejable para una empresa a la hora de guarda y gestionar las imágenes privadas de nuestra empresa

4. Trabajos futuros

Para próximos trabajos podríamos implementar la autentificación de usuarios con un servidor LDAP o Active Directory. También podríamos añadir una interfaz web donde los usuarios accedan y puedan ver las imágenes que están subidas y las diferentes versiones de la imagen.

5. Bibliografía

https://docs.docker.com/registry/