

Deploy a self-hosted registry secured with x509 client certs.

 portainer.io/blog/deploy-a-self-hosted-registry-secured-with-x509-client-certs

by Neil Cresswell, on February 19, 2022

We were recently approached by a user and asked if Portainer supported self-hosted registries that implemented authentication through x509 client certs, and NOT with username/password credentials.

In order to give them an answer, we first wanted to replicate the setup and test it. We found that the process to setup this environment isn't well documented on the internet, so are publishing our steps here, just in case you want to do this too..

Most of this work is executed via the CLI on your host environments, so bear with us..

OK, you need at least 2 VMs for this to work.. one that will host the registry, and one that will act as your client.

Step 1, On the REGISTRY VM (in my case, its Ubuntu). Install Docker.

Follow the instructions here: <https://docs.docker.com/engine/install/ubuntu/>

1. apt-get install ca-certificates curl gnupg lsb-release
2. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
3. echo "deb [arch=\$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
4. apt-get update
5. apt-get install docker-ce docker-ce-cli containerd.io
6. systemctl enable docker
7. docker info (just to check all started ok)

Step 2, create the self-signed SSL cert that will be used for the registry server instance (will actually be used by the nginx proxy). Note you need to have a FQDN for the registry, so make sure you register one accordingly, and then use in the line 4 below. For this blog, i will use registrydemo.portainer.io

1. mkdir -p /opt/registry/
2. cd /opt/registry/
3. mkdir -p certs
4. openssl req -nodes -newkey rsa:8192 -days 365 -x509 -keyout certs/server.key -out certs/server.cert -batch -addext "subjectAltName = DNS:<YOUR FQDN HERE>"

```

root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# openssl req -nodes -newkey rsa:8192 -days 365 -x509 -keyout certs/server
.key -out certs/server.cert -batch -addext "subjectAltName = DNS:registrydemo.portainer.io"
Generating a RSA private key
.....++++
.....++++
writing new private key to 'certs/server.key'
-----
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# █

```

Step 3, create the client CA certificate (the one we will use to generate all client certs that will be used for authentication).

1. `cd /opt/registry/`
2. `openssl req -nodes -newkey rsa:8192 -days 365 -x509 -keyout client-ca.key -out certs/client-ca.cert -batch -subj "/commonName=docker-registry-client-ca"`

```

root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# cd /opt/registry/
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# openssl req -nodes -newkey rsa:8192 -days 365 -x509 -keyout client-ca.ke
y -out certs/client-ca.cert -batch -subj "/commonName=docker-registry-client-ca"
Generating a RSA private key
.....++++
.....++++
writing new private key to 'client-ca.key'
-----
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# █

```

Step 4, generate a client cert using the CA cert above.

1. `openssl genrsa -out client.key 4096`
2. `openssl req -new -key client.key -out client.csr -batch -subj "/commonName=docker-registry-client"`
3. `openssl x509 -req -days 365 -in client.csr -CA /opt/registry/certs/client-ca.cert -CAkey /opt/registry/client-ca.key -set_serial 01 -out client.cert`
4. `rm -f client.csr`

```

-----
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# openssl genrsa -out client.key 4096
openssl req -new -key client.key -out client.csr -batch -subj "/commonName=docker-registry-client"
sudo openssl x509 -req -days 365 -in client.csr -CA /opt/registry/certs/client-ca.cert -CAkey /opt/registry/client-ca.k
ey -set_serial 01 -out client.cert
rm -f client.csrGenerating RSA private key, 4096 bit long modulus (2 primes)
.....++++
.....++++
e is 65537 (0x010001)
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# openssl req -new -key client.key -out client.csr -batch -subj "/commonNa
me=docker-registry-client"
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# sudo openssl x509 -req -days 365 -in client.csr -CA /opt/registry/certs/
client-ca.cert -CAkey /opt/registry/client-ca.key -set_serial 01 -out client.cert
Signature ok
subject=CN = docker-registry-client
Getting CA Private Key
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# rm -f client.csr
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# █

```

Step 5, Configure and start the Docker Registry Container

1. create a config file in `/opt/registry/config.yml`

2. version: 0.1

http:

secret: randomsecretgoeshere

addr: 0.0.0.0:80

storage:

filesystem:

rootdirectory: /var/lib/registry

maxthreads: 100

delete:

enabled: true

3. docker run -d --restart=always --name registry -v

/opt/registry/config.yml:/etc/docker/registry/config.yml registry:2

```
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# docker run -d --restart=always --name registry -v /opt/registry/config.y
ml:/etc/docker/registry/config.yml registry:2
4342e04518b0a375dff76557199ff703a2903c5e8cde9febfa6931b187c7d7fe
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# docker logs 434
time="2022-02-19T21:43:41.675438119Z" level=info msg="redis not configured" go.version=go1.16.13 instance.id=36bba49f-7
74a-4af8-94f9-5241440e1567 version="v2.8.0-beta.1+unknown"
time="2022-02-19T21:43:41.687103944Z" level=info msg="listening on [::]:80" go.version=go1.16.13 instance.id=36bba49f-7
74a-4af8-94f9-5241440e1567 version="v2.8.0-beta.1+unknown"
time="2022-02-19T21:43:41.687360556Z" level=info msg="Starting upload purge in 28m0s" go.version=go1.16.13 instance.id=
36bba49f-774a-4af8-94f9-5241440e1567 version="v2.8.0-beta.1+unknown"
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry#
```

Step 6, Configure and start the nginx proxy Container

1. create a nginx config file in /opt/registry/nginx-registry.conf

2. server {

listen 443 ssl;

keepalive_timeout 70;

client_max_body_size 0;

ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl_ciphers

ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5;

ssl_session_cache shared:SSL:10m;

ssl_session_timeout 10m;

ssl_certificate /etc/nginx/ssl/server.cert;

ssl_certificate_key /etc/nginx/ssl/server.key;

ssl_client_certificate /etc/nginx/ssl/client-ca.cert;

ssl_verify_client on;

location / {

proxy_pass http://registry;

}

}

3. docker run -d --restart=always --name nginx --publish 443:443 --link

registry:registry -v /opt/registry/nginx-registry.conf:/etc/nginx/conf.d/default.conf

-v /opt/registry/certs:/etc/nginx/ssl nginx

```

root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# docker run -d --restart=always --name nginx --publish 443:443 --link registry:registry -v /opt/registry/nginx-registry.conf:/etc/nginx/conf.d/default.conf -v /opt/registry/certs:/etc/nginx/ssl nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
5eb5b503b376: Pull complete
1ae07ab881bd: Pull complete
78091884b7be: Pull complete
091c283c6a66: Pull complete
55de5851019b: Pull complete
b559bad762be: Pull complete
Digest: sha256:2834dc507516af02784808c5f48b7cbe38b8ed5d0f4837f16e78d00deb7e7767
Status: Downloaded newer image for nginx:latest
38be9bb4e7ef7a0d880cc38309d8446a55ec192471a217da218841adea2eb554
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# docker logs 38b
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/02/19 21:48:29 [notice] 1#1: using the "epoll" event method
2022/02/19 21:48:29 [notice] 1#1: nginx/1.21.6
2022/02/19 21:48:29 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/02/19 21:48:29 [notice] 1#1: OS: Linux 5.4.0-97-generic
2022/02/19 21:48:29 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/02/19 21:48:29 [notice] 1#1: start worker processes
2022/02/19 21:48:29 [notice] 1#1: start worker process 30
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry#

```

Step 7, make sure nginx and registry are working.

`curl -v -k --cert ./client.cert --key ./client.key https://localhost:443/v2/`

```

root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry# curl -v -k --cert ./client.cert --key ./client.key https://localhost:443/v2/
* Trying ::1:443...
* TCP_NODELAY set
* Connected to localhost (::1) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
*   CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / AES256-SHA
* ALPN, server accepted to use http/1.1
* Server certificate:
*   subject: C=AU; ST=Some-State; O=Internet Widgits Pty Ltd
*   start date: Feb 19 21:27:23 2022 GMT
*   expire date: Feb 19 21:27:23 2023 GMT
*   issuer: C=AU; ST=Some-State; O=Internet Widgits Pty Ltd
*   SSL certificate verify result: self signed certificate (18), continuing anyway.
> GET /v2/ HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.21.6
< Date: Sat, 19 Feb 2022 21:49:49 GMT
< Content-Type: application/json; charset=utf-8
< Content-Length: 2
< Connection: keep-alive
< Docker-Distribution-API-Version: registry/2.0
<
* Connection #0 to host localhost left intact
root@ubuntu-s-1vcpu-2gb-sfo3-01:/opt/registry#

```

Step 8, now we need to generate the instructions to be used on the Docker Client machines..

1. Change <YOURFQDNHERE> to your registry URL, then Run the following:

2.

```
HOST=registrydemo.portainer.io
SERVER_IP="$(wget -q -O- curlmyip.org)"
cat << __EOF__
set -xe
SERVER_CERT="$(cat /opt/registry/certs/server.cert)"
CLIENT_KEY="$(cat client.key)"
CLIENT_CERT="$(cat client.cert)"
mkdir -p /etc/docker/certs.d/$HOST:443
echo "\$SERVER_CERT" > /etc/docker/certs.d/$HOST:443/ca.crt
echo "\$CLIENT_CERT" > /etc/docker/certs.d/$HOST:443/client.cert
echo "\$CLIENT_KEY" > /etc/docker/certs.d/$HOST:443/client.key
echo "\$(cat /etc/hosts | grep -v '$HOST')" > /etc/hosts
echo "$SERVER_IP $HOST" >> /etc/hosts
__EOF__
```
3. Copy the results to your clipboard and take a note of it.

Step 9, OK, so now SWITCH to your Docker Host VMs (repeat this on every docker host that you want to access this registry).

run the script output from before on the host.

```
+ echo '-----BEGIN RSA PRIVATE KEY-----
MIIJKAIBAAKCAgEAWbfCnnfuTCzvVnj3WZPceoybV6Lf02mM9KGVDiH6Q1XTvSL/
qM+L/ORwYptgKFSs+Qs9/ws/dxBe4B4wQciwgmYHYKKuRG5J4Axya+/72rQTEdyE
hCC6p6WYRBWxa4ABQCCLGOVMcmcaulIQF6LQG/cUVMGFp+LIypUA6ONQFL24toivw
DTf21S9DQgFudWQ3bcQLED/ZDK6no3CGJAJ8j/LhphmGLtqzxNXz/HMoHzCerZ3M
bn8H/8EgGw+3RG/0Bur6ojYN1D/sPJVP2KUXg0AerYU4o4WF1T9qCauHGh06svqI
mrJAJb0kw5gayj11P5siWixcXBiplDg7y2c/tv5jGLUM12Qt3PkWMHxqYPLuO86c
1RY4glJ8GDLzxwNq3bCQLEd/ZDK6no3CGJAJ8j/LhphmGLtqzxNXz/HMoHzCerZ3M
bx+kzTD1KhdbYTW5JTajXVIJ+8iq6/yuVK6f/0LP7EutIp3g3q2MN3qT9H2EPZ6r
+1DI+YS51Lt7Ptjrqs3bLxeBagisYwZwM4HqYwATVuszxhCJ9pB+Ka+2Ud5wTdtm
Chlmcv6UB5y7ZYBKNSnCqWFhLlgnFgc+3uyZ8sZJINgc8oHH2qWXH7fa2UsmHfXy
E+v4Ae0S+EeJudL716ieX5rAaYVh1zXOU7zpJRgdaKuA6Xm6cIyZRx0m7rkCAwEA
AQKCAgEAhUMAozs1t0mCLB5zKjBjKBSCLt29ly6Ft/BRKOhggjc7FPTRDH0ohAU
MAFrGNOHWPudPoJjWmnsmoip8dc5dbrygEvpTjIAqBxn61DhFQajx7nHf410Wjfm
dmxeovEf7DekeK/21eRFXO21kvulg5W8idwgeZk8temFI+rW11YZFmLbdHx3VRh6
UEHjluXmK6Cda8RACDUGJa0W21bLkcsn57fPUxQobtQwUA54H1y3DABtzMdxHUOi
NOjBTlclrEyhtk0xHj4LKKIdx49gqWPGtt/aosML2jMRoy1YdlvzJ4SpzKGrPtZL
IDLW2yGEMwlpjFbQApserW4x1C9m5w9UQQaQ8gi6Nf08fAc1PJ6oTFuRQTsSqYMo
s3LyAQwoFhu2AiQbnQjckHLL74Ulnqg6yZwJBosuVSX9MLRBj6Nyko3+Ti2bti6j
uspkk+ZpgOX7nw+KBqbZwu+lp3BwVe5ayAtiZNmj99cx1c+FfyL2dfRoftpYhKS1
p3Z8sVmHjxusPE13LrzQgaAdK/LBPGypFhXJYTx8LxnVz42Y8Pm8qM1JrI4UOyu0
VL37CxltdtWqjhygs0HaXwGt6yWPZ+KSd6SjhFPsXNK+WJat+elytWhmWLyL9ZR
lezClkykKMUH+qApFaybC8RlQ/alqccRjhcyM9BgWmsPATOMkrECggEBAOSXefbj
+mKn3Cx3NzUVJ5CG+qs3JN8BNK4QkA/ilw+ON/rYmzlj1L+xhz2mJdAh62Qwx2i
i8xoTwoH4v+K0Wj+Yr10xdoALogWK1cuJfjF7HidkAe5GOBLZ/wwzCEwCKhAa2yc
MVy/e3j5kvr93PEu0EufUu0czEqMDmfbjnv1TDA0g6TFyUEo1DZSrutzblWorwMet7
y1Y51bQn+oUMPAIqdf6Tit65yQclMP+lwraxFfJL0+dbBaK2iPZ3FW63/GrQ3SGP
p7tXfXexpIqNwNT7WELsu5hXHf5kDo4IFew8RKZ6NYDWGFIJKusta4VtylRj3UT
mMDMryBzhHiQCq8CggEBANjx2c6XDhdIAOSu0gJ7bdr1PR1I5RcOB7iIBAESDOBg
ahwbybqhDDLfXXuozGNI02MhulHDzVXdtLTKj4VuDbnzASgbU/bVbtXEXf9Jr+bz
NkrITEjMF8GaCYyyAYiZyZf9a+USQiiSpk4GVnLB3iwQ9uWC0DvLiYiL0BmnEWp
+CgidNhmOHPEt6jMvLSbPWqjpr4r6bR9iMjdxh8LYWlZDpp+ievha2RkDAtnTmgz
FivAhtKOSIKBEwrtg+ze+W+pi0cBVH100S8fxyF6ABLCid8+QGztA8hvaUEW4B1
DF+6jq12LsVMGIza5eCG+WMvQeVFb8eCdyd3Lc/TlxcCggEAffjLJhc2QK/oschw
8BvcVdVmWwD6nLp5YYNXIKoNgleJoH8mnoRTW5Vzb/qlVJUHMCSam3Y8PEUXKiv2
eITZqjymAZXK3zWTKyN2dvD85B+OlVmfSeJDRzfG8j+h/pgN7Uqy6IOa4FXVoXFI
AnFsFK2LskoUjLJkjtKpO9LsflG5znJYkBTDC+H4vfl1jmcNfUso93CdOcNddY6i
jDeVykvEAqskWGFh20IvonKivzMPuRr4iAKtNGILUGeHtCBIRkUb16bm9GLEjduk
h0AXEBuYqWZdEOkuoB8z0CGMES09bNbMbUQj6cVsD/95j5n/+hEsUYSfJYcm8ec3
FYmrmQKCAQBRQc0F/zgs6jBDD2N7wIVzr3PCG7aK+xtTN8d7/AnMpPA5nL6V0Zq+
F0GphqD/cxCSKdjitUccqUz9Bn3ZSU5k3qsJPasmYe0DHaxjaVM0WtZ0Di/ceG1R
e4OmKio+4vNKNYaLJlqWXc9inCcXfpxDvrxq/iNGuaJMcDNomMZhpXcoeRB15fx
bI81t3Ha5+aKib/brRD9JKPumHr7wYGKAKjUzj6UljSmP/7u6oPIr2HZUz14OgjY
sMXUKWb14WJmO4fKUao1T57Ra6TkpxctvO2RutJ9m7S0kmG5BDfNlonK9VJoELcV
+RmagUNm5MdWkiYxXN+wR6KfqpKdJeKDAoIBAEoFbQlmce/Gd5kZFrny5JIXAfTg
Ja42vzyemjFCv+mqSWBMFVGbqxsV29GUY2T9iJzLUuP1KteVaFljuOxvY4he+N
nEBH61OjFikLckCEPo4DALWKKu40d0RNpouILS24G61gIt+ayMIzf6v/6s4BxZRn
2/PmjQV4hu0y17YTRKVazUJm39mS7kbDDRSDBA/ZEGFjSw/05Y/yLH0vGEBMftQ
SXz6SKa8egRVqW3nF5TSuTUpkfbnTQUY9H6dab9fnJyuCrHksml3t59+ipfmsRQ
SmHN7CkhUlij40vHZCwpiluUs0p2+qMU3bHS7HgRFB8MDTG+JTPkTxQZ32Q=
-----END RSA PRIVATE KEY-----'
root@democlient:~# echo "$(cat /etc/hosts | grep -v 'registrydemo.portainer.io')" > /etc/hosts
++ grep --color=auto -v registrydemo.portainer.io
++ cat /etc/hosts
+ echo '# Your system has configured '\''manage etc hosts'\'' as True.
# As a result, if you wish for changes to this file to persist
# then you will need to either
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.tmpl
# b.) change or remove the value of '\''manage etc hosts'\'' in
# /etc/cloud/cloud.cfg or cloud-config from user-data
#
127.0.1.1 democlient democlient
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters'
root@democlient:~# echo "137.184.232.252 registrydemo.portainer.io" >> /etc/hosts
+ echo '137.184.232.252 registrydemo.portainer.io'
root@democlient:~#
```

Step 10, Switch to the /etc/docker/certs.d/ directory, and check you have a directory for the FQDN of your registry, and inside that directory are 3x certificate files.


```
+ cd registrydemo.portainer.io:443
root@democlient:/etc/docker/certs.d/registrydemo.portainer.io:443# ls
+ ls --color=auto
ca.crt  client.cert  client.key
root@democlient:/etc/docker/certs.d/registrydemo.portainer.io:443#
```

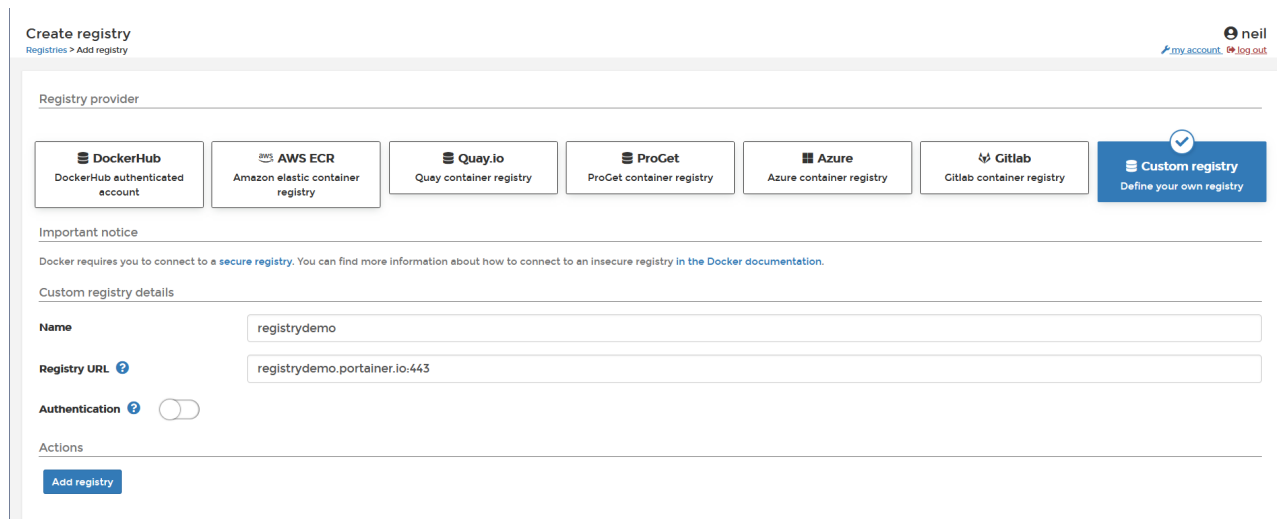
Step 11, you should now be able to use the registry. Test it as follows:

1. docker pull nginx
2. docker tag nginx registrydemo.portainer.io:443/nginx:latest
3. docker push registrydemo.portainer.io:443/nginx:latest
4. docker rmi registrydemo.portainer.io:443/nginx:latest
5. docker rmi nginx
6. docker run -d registrydemo.portainer.io:443/nginx:latest
7. docker image ls

```
root@democlient:~# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
5eb5b503b376: Pull complete
1ae07ab881bd: Pull complete
78091884b7be: Pull complete
091c283c6a66: Pull complete
55de5851019b: Pull complete
b559bad762be: Pull complete
Digest: sha256:2834dc507516af02784808c5f48b7cbe38b8ed5d0f4837f16e78d00deb7e7767
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
root@democlient:~#
root@democlient:~# docker tag nginx registrydemo.portainer.io:443/nginx:latest
root@democlient:~#
root@democlient:~# docker push registrydemo.portainer.io:443/nginx:latest
The push refers to repository [registrydemo.portainer.io:443/nginx]
762b147902c0: Layer already exists
235e04e3592a: Layer already exists
6173b6fa63db: Layer already exists
9a94c4a55fe4: Layer already exists
9a3a6af98e18: Layer already exists
7d0ebbe3f5d2: Layer already exists
latest: digest: sha256:bb129a712c2431ecce4af8dde831e980373b26368233ef0f3b2bae9e9ec515ee size: 1570
root@democlient:~#
root@democlient:~# docker rmi registrydemo.portainer.io:443/nginx:latest
Untagged: registrydemo.portainer.io:443/nginx:latest
Untagged: registrydemo.portainer.io:443/nginx@sha256:bb129a712c2431ecce4af8dde831e980373b26368233ef0f3b2bae9e9ec515ee
root@democlient:~#
root@democlient:~# docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:2834dc507516af02784808c5f48b7cbe38b8ed5d0f4837f16e78d00deb7e7767
Deleted: sha256:c316d5a335a5cf324b0dc83b3da82d7608724769f6454f6d9a621f3ec2534a5a
Deleted: sha256:67e568696593c33b4a15c9d81dc6f67499b8d973b88eb49b53d47bf4dbf4d187
Deleted: sha256:0f8d4e3d979c540644f248b4206cf540978166b095223bdc950628dca2e8f3f1
Deleted: sha256:5d75bfe8a7422476a495b27c8a1598d1206137631d350b8bdee13bc88f365282
Deleted: sha256:8284a9e28c625b2826efdd6160ealff7f710881a4a2afe1ef58a5eb51d3f919e
Deleted: sha256:89a1db9e1079b7574c1a707bc8c1fe04ff723bc71d4bca8bc48653e9a32186d2
Deleted: sha256:7d0ebbe3f5d26c1b5ec4d5dbb6fe3205d7061f9735080b0162d550530328abd6
root@democlient:~#
root@democlient:~# docker run -d registrydemo.portainer.io:443/nginx:latest
Unable to find image 'registrydemo.portainer.io:443/nginx:latest' locally
latest: Pulling from nginx
5eb5b503b376: Pull complete
1ae07ab881bd: Pull complete
78091884b7be: Pull complete
091c283c6a66: Pull complete
55de5851019b: Pull complete
b559bad762be: Pull complete
Digest: sha256:bb129a712c2431ecce4af8dde831e980373b26368233ef0f3b2bae9e9ec515ee
Status: Downloaded newer image for registrydemo.portainer.io:443/nginx:latest
b09d48f96b4593100ba7f435079438f2ae023bc9fc32abab134334600c7dd879
root@democlient:~# docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED      SIZE
registrydemo.portainer.io:443/nginx  latest     c316d5a335a5  3 weeks ago  142MB
root@democlient:~#
```

You now have a private registry, that is secured with HTTPS, and authenticated with client certificates.

Step 12, In Portainer, you simply need to add the registry with no authentication, and Docker will take care of the rest.



Create registry
Registries > Add registry

Registry provider

DockerHub
DockerHub authenticated account

AWS ECR
Amazon elastic container registry

Quay.io
Quay container registry

ProGet
ProGet container registry

Azure
Azure container registry

Gitlab
Gitlab container registry

Custom registry
Define your own registry

Important notice

Docker requires you to connect to a [secure registry](#). You can find more information about how to connect to an insecure registry in the [Docker documentation](#).

Custom registry details

Name
registrydemo

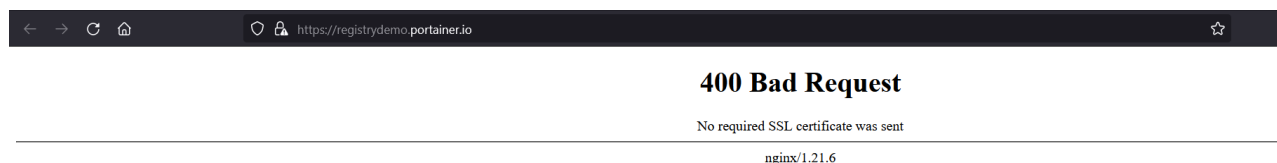
Registry URL
registrydemo.portainer.io:443

Authentication ☐

Actions

Add registry

For proof, try to open your registry FQDN from a browser, and see you cannot access it.



Done.

Hope this is of use to you.

Interested in running Portainer in a business environment?

Portainer Business is our fully featured, fully supported business product. It is used by some of the largest organizations in the world to deliver a powerful self-service container management experience for developers and IT teams. With more than 500,000 active users, Portainer is proven to be the simplest and most effective way of managing Docker, Swarm, and Kubernetes environments.

[GET 5 NODES OF PORTAINER BUSINESS FREE](#)