```
Docker私有仓库使用域名和限制登录
Author：LiNing
```

Docker私有仓库一般使用**Host:Port**形式来代表仓库名称，如果有域名指向这个**Host:Port**，那么我们可以使用域名来代表仓库名称，这样一方面比较容易记忆，另一方面不会轻易暴露Registry使用的IP和端口。

1. 使用域名登录：

前提：假设域名 **myregistrydomain.com** 指向 **xxx.xxx.xxx.xx:5000**，如果不想花钱的话，改本地host。

在Registry服务端：

- 从CA提供商获取证书：

首先，你需要从一个CA提供商获取证书，如果在信任区域内，可以使用自签名证书，使用**openssl**生成。

其中，证书文件和秘钥文件分别为crt和key文件，分别拷贝到/opt/data/certs/domain.crt和/opt/data/certs/domain.key

```
mkdir -p /opt/data/certs
openssl req -newkey rsa:4096 -nodes -sha256 \
    -keyout /opt/data/certs/domain.key -x509 -days 365 \
    -out /opt/data/certs/domain.crt
```

官网上这么说的

Using self-signed certificates

> **Warning**: using this along with basic authentication requires to **also** trust the certificate into the OS cert store for some versions of docker [see below]

This is more secure than the insecure registry solution. You must configure every docker daemon that wants to access your registry

1. Generate your own certificate:
   mkdir -p certs && openssl req
   -newkey rsa:4096 -nodes -sha256 -keyout certs/domain.key
   -x509 -days 365 -out certs/domain.crt
2. Be sure to use the name `myregistrydomain.com` as a CN.
3. Use the result to start your registry with TLS enabled
4. Instruct every docker daemon to trust that certificate.
   This is done by copying the `domain.crt` file to `/etc/docker/certs.d/myregistrydomain.com:5000/ca.crt`.
5. Don't forget to restart the Engine daemon.

使用自签名证书需要注意的一点就是，在设置CN（Common Name）的时候，把要配置的域名写上。

- 运行Docker服务：

```
docker run -d --name registry -p 5000:5000 --restart=always \
    -v /opt/data/certs:/certs \
    -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
    -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
    registry
```

在Registry客户端：

同样需要将服务端的证书**domain.crt**复制到客户端/etc/docker/certs.d/myregistrydomain.com:5000目录下，命名文件为**ca.crt**，其中**myregistrydomain.com:5000**为你注册的域名和使用的端口。

可以使用域名来代表仓库，进行push和pull操作，如：

```
docker tag hello-world myregistrydomain.com:5000/myfirstimage
```

```
docker push myregistrydomain.com:5000/myfirstimage
docker rmi myregistrydomain.com:5000/myfirstimage
docker pull myregistrydomain.com:5000/myfirstimage
```

2. 限制登录：

在Registry服务端：

- 设置用户名和密码。

```
root@fireling-virtual-machine:/# cd opt/data/
root@fireling-virtual-machine:/opt/data# ls
auth  certs  registry
root@fireling-virtual-machine:/opt/data# docker run --name auth_registry --entrypoint htpasswd registry -Bbn testuser testpassword > /opt/data/auth/htpasswd
root@fireling-virtual-machine:/opt/data# cd auth/
root@fireling-virtual-machine:/opt/data/auth# ls
htpasswd
root@fireling-virtual-machine:/opt/data/auth# cat htpasswd
testuser:$2y$05$XltVqWmsEYf461vyRh5GC.ETDGJK001iL9dWOqBLjk0Xvm5nWvrpe

root@fireling-virtual-machine:/opt/data/auth#
```

这里，设置用户名为testuser，密码为testpassword.
可以看到在生成的htpasswd中，保存了用户名和密码的相关信息。

- 运行Docker服务：

```
docker run -d --name registry -p 5000:5000 --restart=always \
    -v /opt/data/auth:/auth \
    -e "REGISTRY_AUTH=htpasswd" 
    -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
    -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
    registry
```

```
root@fireling-virtual-machine:/opt/data/auth# docker run -d --name registry -p 5000:5000 --restart=always -v /opt/data/auth:/auth -e "REGISTRY_AUTH=htpasswd" -e
_HTPASSWD_REALM=Registry Realm" -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd registry
e4fe332a59f1ff3b2db64117d2037d409315e65b0ef5173d0d5badfb0474c606
root@fireling-virtual-machine:/opt/data/auth# docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
e4fe332a59f1        registry            "/entrypoint.sh /etc/"   5 seconds ago       Up 4 seconds        0.0.0.0:5000->5000/tcp   registry
root@fireling-virtual-machine:/opt/data/auth#
```

可以看到服务已经在Linux后台运行，并且服务器的5000端口指向了registry容器的5000端口，这样就可以通过“IP+端口”的方式进行访问了。

在Registry客户端：

采用与登录Docker Hub一样的命令即可，如在服务器上看到的IP为192.168.142.148，则

```
root@fireling-virtual-machine:/home/fireling# ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:22:8e:13:70
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:22ff:fe8e:1370/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:868 errors:0 dropped:0 overruns:0 frame:0
          TX packets:695 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:85139 (85.1 KB)  TX bytes:93566 (93.5 KB)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:8c:39:37
          inet addr:192.168.142.148  Bcast:192.168.142.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe8c:3937/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1027 errors:0 dropped:0 overruns:0 frame:0
          TX packets:735 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:147593 (147.5 KB)  TX bytes:91726 (91.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:520 errors:0 dropped:0 overruns:0 frame:0
          TX packets:520 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:53899 (53.8 KB)  TX bytes:53899 (53.8 KB)

veth753635f Link encap:Ethernet  HWaddr 8e:dc:08:84:93:c4
          inet6 addr: fe80::8cdc:8ff:fe84:93c4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:648 (648.0 B)  TX bytes:4671 (4.6 KB)

root@fireling-virtual-machine:/home/fireling#
```

输入命令：

```
docker login 192.168.142.148:5000
```

会提示输入用户名和密码，输入正确则登录成功，接下来就可以进行push和pull操作了。

```
fireling@fireling-virtual-machine:~$ docker login 192.168.142.148:5000
Username (testuser): testuser
Password:
Login Succeeded
fireling@fireling-virtual-machine:~$
```