How to Setup Private Docker Registry on Ubuntu 18.04 LTS

Docker Registry or 'Registry' is an open source and highly scalable server-side application that can be used to store and distribute Docker images. It was a server-side application behind the Docker Hub. In most use cases, a Docker Registry is a great solution if you want to implement the CI/CD system on your application development. The Private Docker Registry gives more performances for the development and production cycle by centralizing all your custom Docker images of application in one place.

On this page

In this tutorial, we're going to show you how to install and configure a Private Docker Registry on a Ubuntu 18.04 server. We will use an Nginx web server and protect the Registry with a username and password (basic auth).

Prerequisites

- Ubuntu 18.04 server
- Root privileges

What we will do?

- Install Dependencies
- Install Docker and Docker-compose
- Setup Private Docker Registry
- Testing

Step 1 - Install Package Dependencies

First of all, we're going to install some packages dependencies for deploying the Private Docker Registry.

Install packages dependencies using the following command.

```
sudo apt install -y gnupg2 pass apache2-utils httpie
```

The gnupg2 and pass packages will be used to store the password authentication to the docker registry. And the apache2-utils will be used to generate the basic authentication, and httpie will be used for testing.

Step 2 - Install Docker and Docker-compose

Now we're going to install the docker and docker-compose from the official Ubuntu repository.

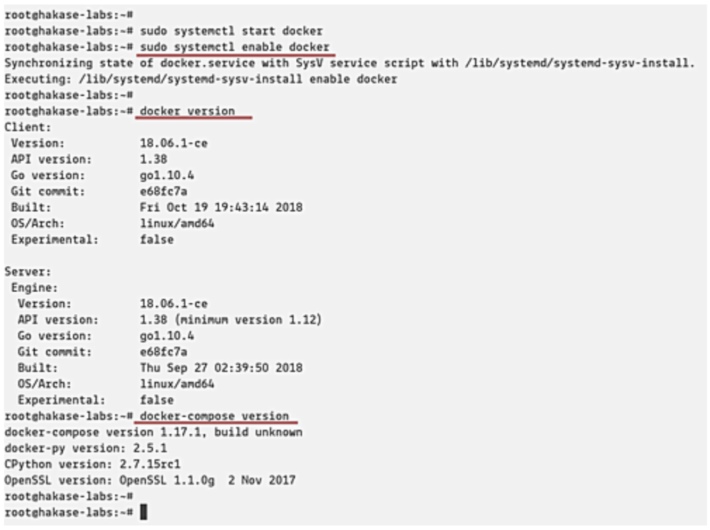
Install Docker and Docker-compose by running the following command.

```
sudo apt install -y docker.io docker-compose -y
sudo systemctl start docker
sudo systemctl enable docker
```

The Docker is up and running, and the Docker-compose has been installed. Check using the command below.

```
docker version
docker-compose version
```

And you will be displayed version of Docker and Docker-compose installed on your system.



```
root@hakase-labs:~#
root@hakase-labs:~# sudo systemctl start docker
root@hakase-labs:~# sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
root@hakase-labs:~#
root@hakase-labs:~# docker version
Client:
 Version:           18.06.1-ce
 API version:       1.38
 Go version:        go1.10.4
 Git commit:        e68fc7a
 Built:             Fri Oct 19 19:43:14 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.1-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.4
  Git commit:       e68fc7a
  Built:            Thu Sep 27 02:39:50 2018
  OS/Arch:          linux/amd64
  Experimental:     false
root@hakase-labs:~# docker-compose version
docker-compose version 1.17.1, build unknown
docker-py version: 2.5.1
CPython version: 2.7.15rc1
OpenSSL version: OpenSSL 1.1.0g  2 Nov 2017
root@hakase-labs:~#
root@hakase-labs:~# 
```

Step 3 - Setup Private Docker Registry

In this step, we're going to configure the Docker Registry environment by creating some directories environment, and create some configuration including the docker-compose.yml, nginx virtual host and additional configuration etc.

- Create Project Directories

Create a new directory for the project called 'registry' and create the 'nginx' and 'auth' directories inside.

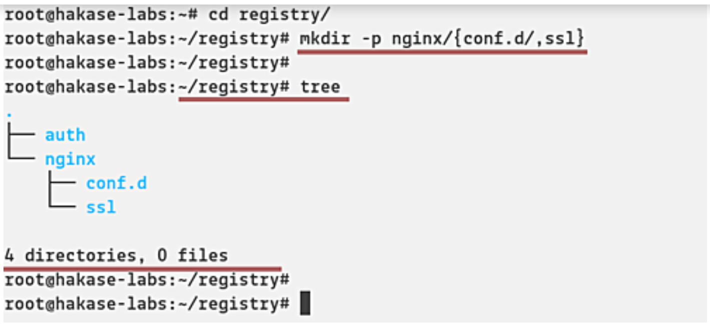
```
mkdir -p registry/{nginx,auth}
```

After that, go to the directory 'registry' and create new directories again inside 'nginx'.

```
cd registry/
mkdir -p nginx/{conf.d/,ssl}
```

And as a result, the project directories look like the following picture.

```
tree
```



```
root@hakase-labs:~# cd registry/
root@hakase-labs:~/registry# mkdir -p nginx/{conf.d/,ssl}
root@hakase-labs:~/registry#
root@hakase-labs:~/registry# tree
.
├── auth
└── nginx
    ├── conf.d
    └── ssl

4 directories, 0 files
root@hakase-labs:~/registry#
root@hakase-labs:~/registry#
```

- Create Docker-compose Script

Now we want to create a new docker-compose.yml script for deploying the Docker Registry.

Go to the 'registry' directory and create a new configuration file 'docker-compose.yml'.

```
cd registry/
vim docker-compose.yml
```

Firstly, define the compose version that you want to use and the service.

```
version: '3'
services:
```

After that, add the first service named 'registry'. The Docker Registry service will be using the docker image that's provided by docker team 'registry:2. It will mount the docker volume 'registrydata' and the local directory named 'auth' that contains basic authentication file 'registry.passwd'. And the last, it will run on the custom docker image named 'mynet' and expose the port 5000 on both container and host.

```
#Registry
  registry:
    image: registry:2
    restart: always
    ports:
    - "5000:5000"
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry-Realm
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.passwd
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - registrydata:/data
      - ./auth:/auth
    networks:
      - mynet
```

Next, the configuration of 'nginx' service that will run HTTP and HTTPS ports and mount the local directory 'conf.d' for virtual host configuration, and the 'ssl' for ssl certificates.

```
#Nginx Service
  nginx:
    image: nginx:alpine
    container_name: nginx
    restart: unless-stopped
    tty: true
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/conf.d/:/etc/nginx/conf.d/
      - ./nginx/ssl/:/etc/nginx/ssl/
    networks:
      - mynet
```

```
#Docker Networks
networks:
  mynet:
    driver: bridge
#Volumes
volumes:
  registrydata:
    driver: local
```

Save and close the configuration.

Below is the complete configuration:

```
version: '3'
services:

#Registry
  registry:
    image: registry:2
    restart: always
    ports:
    - "5000:5000"
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry-Realm
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.passwd
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - registrydata:/data
      - ./auth:/auth
    networks:
      - mynet

#Nginx Service
  nginx:
    image: nginx:alpine
    container_name: nginx
    restart: unless-stopped
    tty: true
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/conf.d/:/etc/nginx/conf.d/
      - ./nginx/ssl/:/etc/nginx/ssl/
    networks:
      - mynet

#Docker Networks
networks:
  mynet:
    driver: bridge
#Volumes
volumes:
  registrydata:
    driver: local
```

- Configure Nginx Virtual Host

After creating the docker-compose script, we will create the virtual host and additional configuration for the nginx service.

Go to 'nginx/conf.d/' directory and create a new virtual host file called 'registry.conf'.

```
cd nginx/conf.d/
vim registry.conf
```

Paste the following configuration.

```
upstream docker-registry {
    server registry:5000;
    server_name registry.hakase-labs.io;
    return 301 https://registry.hakase-labs.io$request_uri;
}

server {
    listen 443 ssl http2;
    server_name registry.hakase-labs.io;

    ssl_certificate /etc/nginx/ssl/fullchain.pem;
    ssl_certificate_key /etc/nginx/ssl/privkey.pem;

    # Log files for Debug
    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;

    location / {
        # Do not allow connections from docker 1.5 and earlier
        # docker pre-1.6.0 did not properly set the user agent on ping, catch "Go *" user agents
        if ($http_user_agent ~ "^(docker\/1\.(3|4|5(?!\.[0-9]-dev))|Go ).*$" )  {
            return 404;
        }

        proxy_pass                          http://docker-registry;
        proxy_set_header  Host              $http_host;
        proxy_set_header  X-Real-IP         $remote_addr;
        proxy_set_header  X-Forwarded-For   $proxy_add_x_forwarded_for;
        proxy_set_header  X-Forwarded-Proto $scheme;
        proxy_read_timeout                  900;
    }

}
```

Save and close.

Next, create an additional configuration to increase the max_body_size on nginx. This will allow you to upload docker images with max size 2GB.

```
vim additional.conf
```

Paste configuration below.

```
client_max_body_size 2G;
```

Save and close.

- Configure SSL Certificate and Basic Authentication

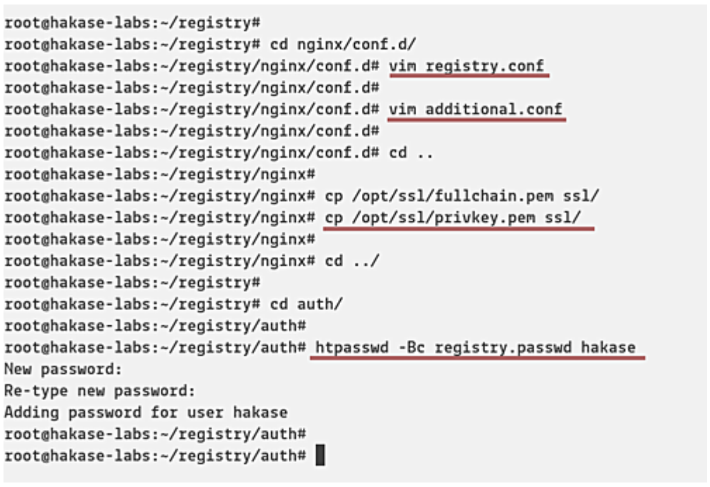
Copy SSL certificate files of your domain to the 'ssl' directory.

```
cp /path/to/ssl/fullchain.pem ssl/
cp /path/to/ssl/privkey.pem ssl/
```

Now go to the 'auth' directory and generate the new password file 'registry.passwd'.

Generate a new password for user hakase.

```
htpasswd -Bc registry.passwd hakase
TYPE THE STRONG PASSWORD
```

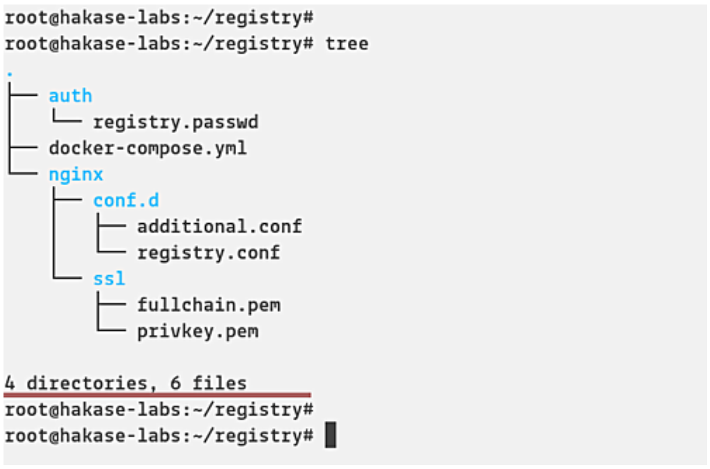


```
root@hakase-labs:~/registry#
root@hakase-labs:~/registry# cd nginx/conf.d/
root@hakase-labs:~/registry/nginx/conf.d# vim registry.conf
root@hakase-labs:~/registry/nginx/conf.d#
root@hakase-labs:~/registry/nginx/conf.d# vim additional.conf
root@hakase-labs:~/registry/nginx/conf.d#
root@hakase-labs:~/registry/nginx/conf.d# cd ..
root@hakase-labs:~/registry/nginx#
root@hakase-labs:~/registry/nginx# cp /opt/ssl/fullchain.pem ssl/
root@hakase-labs:~/registry/nginx# cp /opt/ssl/privkey.pem ssl/
root@hakase-labs:~/registry/nginx#
root@hakase-labs:~/registry/nginx# cd ../
root@hakase-labs:~/registry#
root@hakase-labs:~/registry# cd auth/
root@hakase-labs:~/registry/auth#
root@hakase-labs:~/registry/auth# htpasswd -Bc registry.passwd hakase
New password:
Re-type new password:
Adding password for user hakase
root@hakase-labs:~/registry/auth#
root@hakase-labs:~/registry/auth#
```

And the environment setup for deploying Private Docker Registry has been completed.

Below is the screenshot of our environment files and directories.

```
tree
```



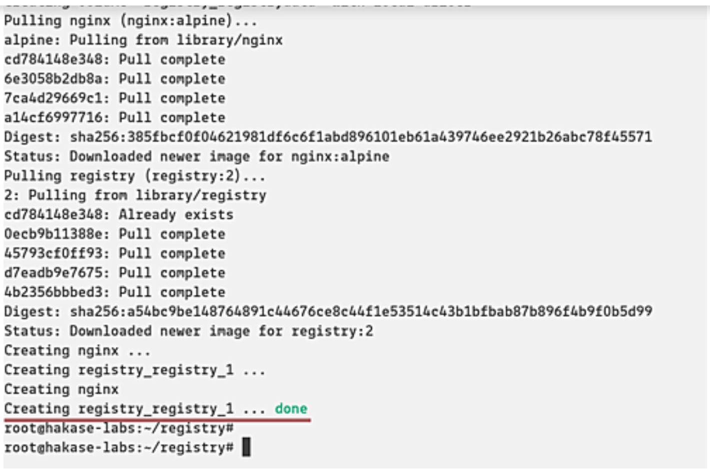
```
root@hakase-labs:~/registry#
root@hakase-labs:~/registry# tree
.
├── auth
│   └── registry.passwd
├── docker-compose.yml
└── nginx
    ├── conf.d
    │   ├── additional.conf
    │   └── registry.conf
    └── ssl
        ├── fullchain.pem
        └── privkey.pem

4 directories, 6 files
root@hakase-labs:~/registry#
root@hakase-labs:~/registry#
```

- Run Docker Registry

Run the Docker Registry using the docker-compose command below.

```
docker-compose up -d
```

And you will get the result as below.

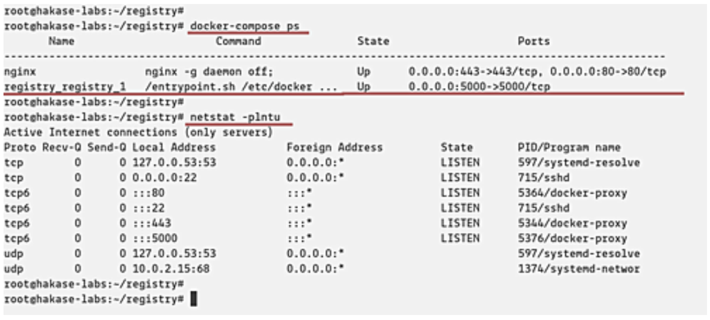


```
Pulling nginx (nginx:alpine)...
alpine: Pulling from library/nginx
cd784148e348: Pull complete
6e3058b2db8a: Pull complete
7ca4d29669c1: Pull complete
a14cf6997716: Pull complete
Digest: sha256:385fbcf0f04621981df6c6f1abd896101eb61a439746ee2921b26abc78f45571
Status: Downloaded newer image for nginx:alpine
Pulling registry (registry:2)...
2: Pulling from library/registry
cd784148e348: Already exists
0ecb9b11388e: Pull complete
45793cf0ff93: Pull complete
d7eadb9e7675: Pull complete
4b2356bbbed3: Pull complete
Digest: sha256:a54bc9be148764891c44676ce8c44f1e53514c43b1bfbab87b896f4b9f0b5d99
Status: Downloaded newer image for registry:2
Creating nginx ...
Creating registry_registry_1 ...
Creating nginx
Creating registry_registry_1 ... done
root@hakase-labs:~/registry#
root@hakase-labs:~/registry#
```

After that, make sure the registry and nginx service is up and running. Check using the following command.

```
docker-compose ps
netstat -plntu
```

And you will be shown the 'registry' service is running on port '5000', and the 'nginx' service will expose the HTTP and HTTPS ports as below.



```
root@hakase-labs:~/registry#
root@hakase-labs:~/registry# docker-compose ps
       Name                     Command               State                   Ports
-------------------------------------------------------------------------------------------------
nginx                 nginx -g daemon off;            Up      0.0.0.0:443->443/tcp, 0.0.0.0:80->80/tcp
registry_registry_1   /entrypoint.sh /etc/docker ...  Up      0.0.0.0:5000->5000/tcp
root@hakase-labs:~/registry#
root@hakase-labs:~/registry# netstat -plntu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      597/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      715/sshd
tcp6       0      0 :::80                   :::*                    LISTEN      5364/docker-proxy
tcp6       0      0 :::22                   :::*                    LISTEN      715/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      5344/docker-proxy
tcp6       0      0 :::5000                 :::*                    LISTEN      5376/docker-proxy
udp        0      0 127.0.0.53:53           0.0.0.0:*                           597/systemd-resolve
udp        0      0 10.0.2.15:68            0.0.0.0:*                           1374/systemd-networ
root@hakase-labs:~/registry#
root@hakase-labs:~/registry#
```

Step 4 - Testing

Before we test our Private Docker Registry, we need to add the Root CA certificate to the docker itself and to the system.

If you're using the pem file certificate, export it to the .crt file using the OpenSSL command.

```
openssl x509 -in rootCA.pem -inform PEM -out rootCA.crt
```

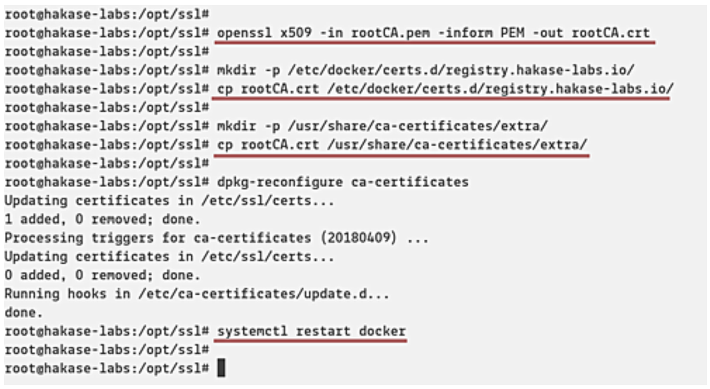
Now create a new directory for docker certificate and copy the Root CA certificate into it.

```
mkdir -p /etc/docker/certs.d/registry.hakase-labs.io/
cp rootCA.crt /etc/docker/certs.d/registry.hakase-labs.io/
```

```
mkdir -p /usr/share/ca-certificates/extra/
cp rootCA.crt /usr/share/ca-certificates/extra/
```

After that, reconfigure the 'ca-certificate' package and restart the Docker service.

```
dpkg-reconfigure ca-certificates
systemctl restart docker
```



```
root@hakase-labs:/opt/ssl#
root@hakase-labs:/opt/ssl# openssl x509 -in rootCA.pem -inform PEM -out rootCA.crt
root@hakase-labs:/opt/ssl#
root@hakase-labs:/opt/ssl# mkdir -p /etc/docker/certs.d/registry.hakase-labs.io/
root@hakase-labs:/opt/ssl# cp rootCA.crt /etc/docker/certs.d/registry.hakase-labs.io/
root@hakase-labs:/opt/ssl#
root@hakase-labs:/opt/ssl# mkdir -p /usr/share/ca-certificates/extra/
root@hakase-labs:/opt/ssl# cp rootCA.crt /usr/share/ca-certificates/extra/
root@hakase-labs:/opt/ssl#
root@hakase-labs:/opt/ssl# dpkg-reconfigure ca-certificates
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Processing triggers for ca-certificates (20180409) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
root@hakase-labs:/opt/ssl# systemctl restart docker
root@hakase-labs:/opt/ssl#
root@hakase-labs:/opt/ssl#
```

- Download Docker Image

Download new Docker image using the following command.

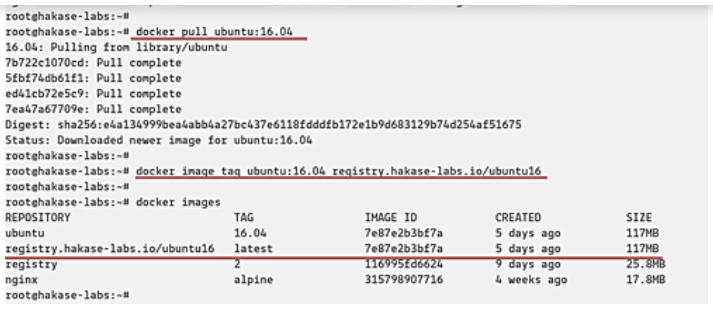
```
docker pull ubuntu:16.04
```

When it's complete, tag the image for the private registry with the command below.

```
docker image tag ubuntu:16.04 registry.hakase-labs.io/ubuntu16
```

Check again the list of Docker images on the system and you will get new images as below.

```
docker images
```



```
root@hakase-labs:~#
root@hakase-labs:~# docker pull ubuntu:16.04
16.04: Pulling from library/ubuntu
7b722c1070cd: Pull complete
5fbf74db61f1: Pull complete
ed41cb72e5c9: Pull complete
7ea47a67709e: Pull complete
Digest: sha256:e4a134999bea4abb4a27bc437e6118fdddfb172e1b9d683129b74d254af51675
Status: Downloaded newer image for ubuntu:16.04
root@hakase-labs:~#
root@hakase-labs:~# docker image tag ubuntu:16.04 registry.hakase-labs.io/ubuntu16
root@hakase-labs:~#
root@hakase-labs:~# docker images
REPOSITORY                        TAG       IMAGE ID       CREATED       SIZE
ubuntu                            16.04     7e87e2b3bf7a   5 days ago    117MB
registry.hakase-labs.io/ubuntu16  latest    7e87e2b3bf7a   5 days ago    117MB
registry                          2         116995fd6624   9 days ago    25.8MB
nginx                             alpine    315798907716   4 weeks ago   17.8MB
root@hakase-labs:~#
```

- Push Image to Private Local Registry

Log in to the Private Docker Registry using the following command.

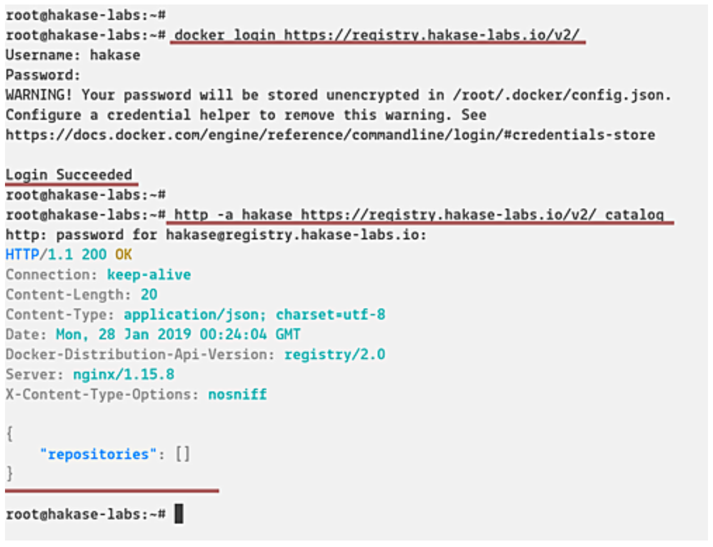
```
docker login https://registry.hakase-labs.io/v2/
```

Type the username and password based on the 'registry.htpasswd' file.

Now check the available of docker image on the Registry.

```
http -a hakase https://registry.hakase-labs.io/v2/_catalog
```

And there is no docker image on the Registry.



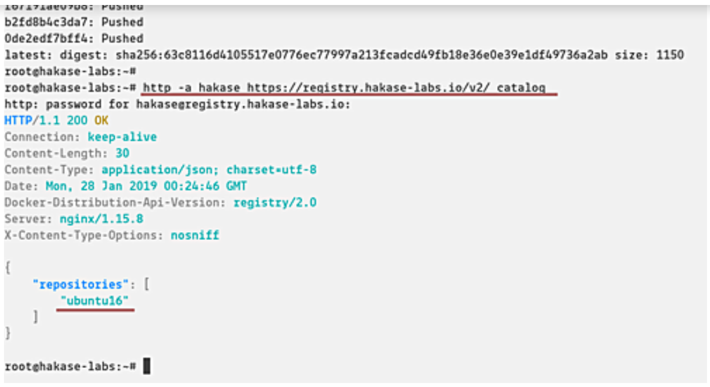
```
root@hakase-labs:~#
root@hakase-labs:~# docker login https://registry.hakase-labs.io/v2/
Username: hakase
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@hakase-labs:~#
root@hakase-labs:~# http -a hakase https://registry.hakase-labs.io/v2/_catalog
http: password for hakase@registry.hakase-labs.io:
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 20
Content-Type: application/json; charset=utf-8
Date: Mon, 28 Jan 2019 00:24:04 GMT
Docker-Distribution-Api-Version: registry/2.0
Server: nginx/1.15.8
X-Content-Type-Options: nosniff

{
    "repositories": []
}

root@hakase-labs:~#
```

Now push our custom image to the Private Docker Registry.

```
docker push registry.hakase-labs.io/ubuntu16
```

Check again and make sure you get the 'ubuntu16' docker image on the Private Repository.

```
http -a hakase https://registry.hakase-labs.io/v2/_catalog
```



```
b2fd8b4c3da7: Pushed
0de2edf7bff4: Pushed
latest: digest: sha256:63c8116d4105517e0776ec77997a213fcadcd49fb18e36e0e39e1df49736a2ab size: 1150
root@hakase-labs:~#
root@hakase-labs:~# http -a hakase https://registry.hakase-labs.io/v2/_catalog
http: password for hakase@registry.hakase-labs.io:
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 30
Content-Type: application/json; charset=utf-8
Date: Mon, 28 Jan 2019 00:24:46 GMT
Docker-Distribution-Api-Version: registry/2.0
Server: nginx/1.15.8
X-Content-Type-Options: nosniff

{
    "repositories": [
        "ubuntu16"
    ]
}

root@hakase-labs:~#
```

And finally, the installation and configuration of Private Docker Registry with Nginx and Basic Authentication has been completed successfully.

About Muhammad Arul

Muhammad Arul is a freelance system administrator and technical writer. He is working with Linux Environments for more than 5 years, an Open Source enthusiast and highly motivated on Linux installation and troubleshooting. Mostly working with RedHat/CentOS Linux and Ubuntu/Debian, Nginx and Apache web server, Proxmox, Zimbra Administration, and Website Optimization. Currently learning about OpenStack and Container Technology.

view as pdf | print

Share this page:

Recommend Tweet Follow G+

Suggested articles

8 Comment(s)

p

I'm not a robot
reCAPTCHA
Privacy - Terms

Submit comment

Comments

By: TiTex **at:** *2019-02-09 06:44:09* Reply

what exactly are you trying to do here ? , you input a certificate and output the same certificate with a different extension of the file.
what you are doing there could be done with `mv rootCA{.pem,.crt}` , or copy
openssl x509 -in rootCA.pem -inform PEM -out rootCA.crt

By: emma **at:** *2019-02-09 10:51:21* Reply

Its just convert the PEM certificate file to the CRT file.
You can see the difference below.
https://crypto.stackexchange.com/questions/43697/what-is-the-difference-between-pem-csr-key-and-crt

By: TiTex **at:** *2019-02-11 18:18:22* Reply

i don't think that's right :)
i think what you wanted to say , is that a file with a .pem extension can contain any data like a ssh private/public key , a PKCS#8 private key (the private counterpart of an ssl cert) or public key (digital/ssl certificate)
also these days we only use X509v3 , so there's nothing to convert to/from , only need to convert from DER to PEM (base64) from time to time, and the reverse
so in the command mentioned above , he clearly doesn't use the -outform parameter which means that it's clearly a base64 encoded public key (ssl certificate) and the outform is also PEM by default, so he only changes it's extension from .pem to .crt , it doesn't make any changes to it's contents or structure
that being said , it's a good howto , thank you! :)

By: emma **at:** *2019-02-11 22:53:54* Reply

Thanks for your comment, I really appreciate it.
Btw, this root certificate is generated with 'mkcert' tool for local domain '*.hakase-labs.io'.
Simply put, I just want to add the RootCA.pem certificate to the '/usr/share/ca-certificates' directory, which is provided by the package 'ca-certificates'. I need to do that so I can check the private registry 'URL' from terminal and web browser/using httpie without any warnings.
So, when I check to the directory '/usr/share/ca-certificates', I found all CA certificates inside has a '.crt' format. And it just come to my mind to convert my 'pem' root certificate to the '.crt' format.

By: TiTex **at:** *2019-02-12 16:41:11* Reply

that's correct update-ca-certificates script looks for files with a .crt extension , you can modify it though with a diffrent name so it doesn't get overwritten by updates and use that :)
but if you have a base64 encoded (PEM) certificate, you don't need to run it through `openssl x509` , you can just copy it `cp rootCA.pem /path/where/to/copy/rootCA.crt` considering that you don't have a file with that name already and don't overwrite something that you need
also i would recommend copy your custom/personal CA certs in /usr/local/share/ca-certificates/ instead of /usr/share/ca-certificates/

By: Sternrassler **at:** *2019-03-14 16:50:55* Reply

Hello, Muhamad,

your manual worked wonderfully compared to the many others.

Hi,
Would you have an idea why when I do
docker login $URL I received a Error 403
Error response from daemon: login attempt to https://$HOST/v2/ failed with status: 403 Forbidden
While if I do :
http -a $USER https://$HOST/v2/_cataloghttp: password for $USER@$HOST: HTTP/1.1 200 OKConnection: keep-aliveContent-Length: 20Content-Type: application/json; charset=utf-8Date: Fri, 15 Mar 2019 10:35:03 GMTDocker-Distribution-Api-Version: registry/2.0Server: nginx/1.15.9X-Content-Type-Options: nosniff
{"repositories": []}
It works !

By: Anurag **at:** *2019-06-12 07:02:06* Reply

I got this error
Error response from daemon: Get https://<IP>/v2/: x509: cannot validate certificate for <IP> because it doesn't contain any IP SANs
Could you please tell the solution?
Thank you

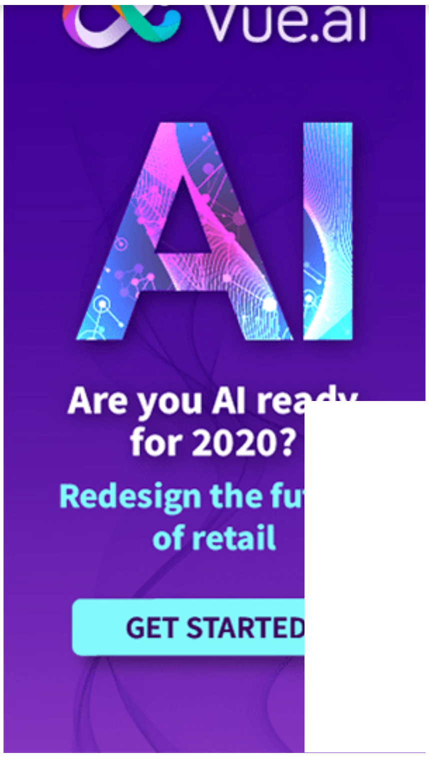
Sign up now!

Tutorial Info

Author:	Muhammad Arul
Published:	Feb 08, 2019
Tags:	linux, nginx, server, ubuntu, virtualization

Share This Page

Recommend Tweet Follow Y
40.2k Followers



Popular Tutorials

How to use grep to search for strings in files on the shell

How to create Docker Images with a Dockerfile

Vim Editor Basics

How to use the Linux ftp command to up- and download files on the shell

How to search files from the Terminal on Linux

The Perfect Server - Ubuntu 18.04 (Bionic Beaver) with Apache, PHP, MySQL, PureFTPD, BIND, Postfix, Dovecot and ISPConfig 3.1

Setting, Changing and Resetting MySQL Root Passwords

How to Setup ZSH and Oh-my-zsh on Linux

Linux Basics - Set a Static IP on Ubuntu

How To Configure Remote Access To Your Ubuntu Desktop