Creating private docker registry with basic authentication on ubuntu

This documentation provides you the steps to create a private docker registry with a basic authentication where you will be given a username and password credentials to login into the registry. Once you are logged in, you can push, pull and view the images inside the private docker registry. We use docker-compose plugin to run the registry container and apache2-utils package to create an authentication setup.

STEP 1:

Installing docker and docker compose

Docker installation:

Update your existing packages:

```
sudo apt update
```

Install a prerequisite package which let apt utilize HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Add GPG key for the official Docker repo to the Ubuntu system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Add the Docker repo to APT sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

Update the database with the Docker packages from the added repo:

```
sudo apt update
```

Install Docker software:

```
sudo apt install docker-ce
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. To verify:

```
sudo systemctl status docker
```

NOTE: To avoid using sudo for docker activities, add your username to the Docker Group

sudo usermod -aG docker ${USER}

Docker-compose installation:

The command below will download the 1.28.5 release and save the executable at /usr/local/bin/docker-compose which will make this software globally accessible as docker-compose:

sudo curl -L "https://github.com/docker/compose/releases/download/1.28.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Set permissions so that the docker-compose file is executable

sudo chmod +x /usr/local/bin/docker-compose

Verify that the installation was successful

docker-compose --version

STEP 2:

Creating docker-compose file with the registry container specification

Create a directory called docker-registry and store the configuration on the main server

mkdir ~/docker-registry

cd ~/docker-registry

Create a sub directory called data inside docker-registry, where the private registry will store its images.

mkdir data

create and open a file called docker-compose.yaml inside docker-registry directory and add the following configurations.

DOCKER COMPOSE FILE:

```
version: '3'
services:
  registry:
    image: registry:2
    ports:
    - "5000:5000"
    environment:
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - ./data:/data
    networks:
      - docker-registry
  docker-registry-host-web-ui:
    image: konradkleine/docker-registry-frontend:v2
    environment:
      ENV_DOCKER_REGISTRY_HOST: registry
      ENV_DOCKER_REGISTRY_PORT: 5000
    ports:
      - 8080:80
    networks:
      - docker-registry
networks:
  docker-registry:
```

1. Defining and mentioning the version of the schema
2. Name the first service registry.
3. Set its image to registry, version 2 in registry service.
4. Under ports, you map the port 5000 on the host to the port 5000 of the container. This allows you to send a request to port 5000 on the server, and have the request forwarded to the registry.
5. In the environment section, you set the REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY variable to /data, specifying in which volume it should store its data.

6. In the volumes section, you map the /data directory on the host file system to /data in the container. The data about images will actually be stored on the host's file system.
7. Similar way create another service for frontend ui , name it docker-registry-host-web-ui and specify the image.
8. Set two environment variables ENV_DOCKER_REGISTRY_HOST and ENV_DOCKER_REGISTRY_PORT
9. In ENV_DOCKER_REGISTRY_HOST specify the service name of the registry.
10. In ENV_DOCKER_REGISTRY_PORT specify 5000 which is the port number.
11. For binding the network specify a field called network.

Save the file and run the docker compose file.

docker-compose up #If docker compose file name is docker-compose.yaml

docker-compose -f <docker_compose_file_name>.yaml up

In a browser window, navigate to your domain and access the v2 endpoint.

http://ip_address:port/v2

http://10.11.100.86:5000/v2

you will find a empty repository initially.

STEP 3:

Setting Up Authentication

obtain the htpasswd utility by installing the apache2-utils package.

sudo apt install apache2-utils -y

Create a subdirectory inside docker-registry for storing the authentication credentials.

mkdir ~/docker-registry/auth

cd ~/docker-registry/auth

Create the first user, replacing username with the username you want to use. The -B flag orders the use of the bcrypt algorithm, which Docker requires:

htpasswd -Bc registry.password username

Enter the password when prompted, and the combination of credentials will be appended to registry.password

Note : To add more users, re-run the previous command without -c which creates a new file:

htpasswd -B registry.password username

After setting credentials , we need to update the docker-compose.yaml file with the following fields for the docker to use the authentication file.

cd ~/docker-registry/

vi docker-compose.yaml

docker-compose.yaml

```
version: '3'
services:
  registry:
    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
    networks:
      - docker-registry
  docker-registry-host-web-ui:
    image: konradkleine/docker-registry-frontend:v2
    environment:
```

ENV_DOCKER_REGISTRY_HOST: registry

ENV_DOCKER_REGISTRY_PORT: 5000

ports:

- 8080:80

networks:

- docker-registry

networks:

docker-registry:

1. Here we are specifying the use of HTTP authentication and provided the path to the file htpasswd created.
2. For REGISTRY_AUTH, you have specified htpasswd as its value, which is the authentication scheme you are using, and set REGISTRY_AUTH_HTPASSWD_PATH to the path of the authentication file.
3. REGISTRY_AUTH_HTPASSWD_REALM signifies the name of htpasswd realm.
4. Mount the ./auth directory to make the file available inside the registry container.
5. Save and close the file.

You can now verify that your authentication works correctly.

cd ~/docker-registry

docker-compose up

In your browser, refresh the page of your domain. You'll be asked for a username and password.

STEP 4:

Push and pull images to the private docker regsitry

On the main server, log in with the username and password you set up previously:

docker login http://your_domain\

example: docker login http://10.11.100.86:5000

Tag your existing image with the domain name as following

Example: docker pull nginx:latest

docker tag nginx:latest 10.11.100.86:5000/my-nginx:latest

Now your image tag will be changes and now try to push it to the private docker registry

docker push 10.11.100.86:5000/my-nginx:latest

Image will be pushed successfully.

Now for testing delete the image 10.11.100.86:5000/my-nginx:latest from your local repository and tr to pull it from private docker registry by following commands

docker pull 10.11.100.86:5000/my-nginx:latest

Now that you've tested pushing and pulling images, you've finished setting up a secure registry that you can use to store custom images.

Since we are not enabled secured (https) connection we need to update mention the ip of our main server as insecure registry in the each node where we need to work with our private docker registry. Following are the steps to update it.

Creating Insecure Registry:

cd ~/etc/docker

Create a daemon.json file if it doesnot exist.
Add the below content into it.

```
{
    "insecure-registries" : ["domain-name:port"]
}
```

Example:

```
{
    "insecure-registries" : ["10.11.100.86:5000"]
}
```

Save and exit the file .
Restart the docker for applying the changes.

systemctl daemon-reload
systemctl restart docker