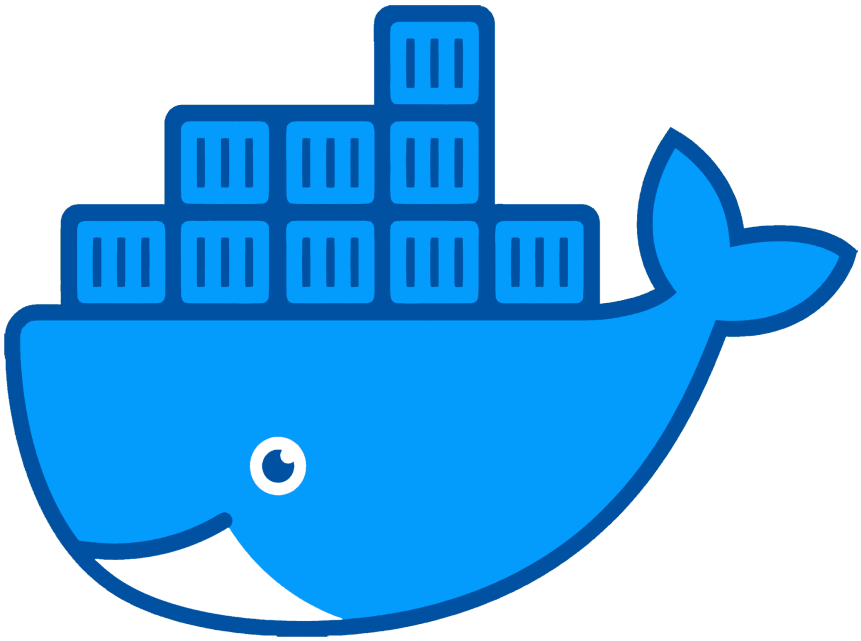


Docker Workshop

ISC 629: Computing Ecosystems
School of Computing
University of South Alabama
August 30, 2022

Conducted by Harith Warnakulasooriya

What is Docker?

“Docker is an open-source platform that enables developers to build, deploy, run, update, and manage containers—standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. “

(I.C., 2022)

Terminologies

1. DockerFile

DockerFile is a root level document that enables the developer to provide instructions on how to build the docker container image.

2. Docker Image

Docker image is the software piece that has all the source code, relevant libraries, and the tools and dependencies to run the docker container. Docker image can be mentioned as a blueprint of a docker container.

3. Docker Container

The Docker container is the running instance of the docker image. This is a virtualized computing instance users can interact.

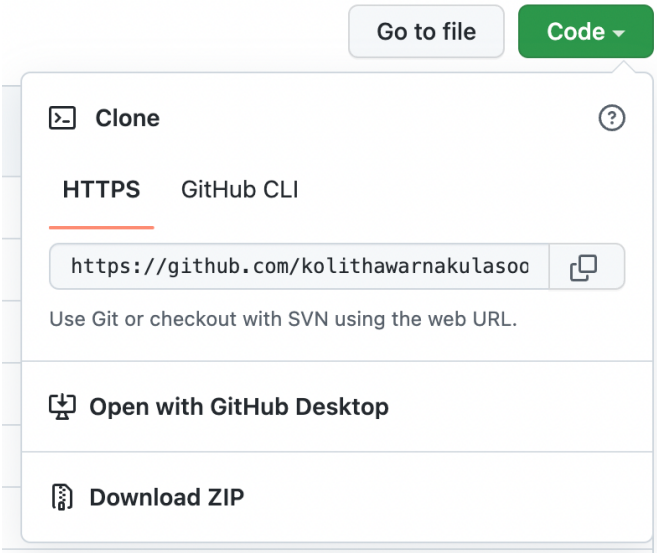
4. Docker Hub

Docker hub is the public repository that has sharable docker images. Any user can publish their docker image on this platform.

Education, I. C. (2022, June 17). Docker. IBM Cloud. https://www.ibm.com/cloud/learn/docker

Requirements for this workshop

1. You must install “Docker” on your computer. Download and install docker using
    a. Windows https://docs.docker.com/desktop/install/windows-install/
    b. Mac https://docs.docker.com/desktop/install/mac-install/
    c. Linux https://docs.docker.com/desktop/install/linux-install/
2. You must install a code editor to edit docker file. We use “Visual Studio Code” editor in this workshop. You can download and install VScode using https://code.visualstudio.com/
3. We use a web application to run in a docker container. You can find the web application at https://github.com/kolithawarnakulasooriya/Docker-Workshop.
    a. If you are using git, you can use this command to clone the repository using the below command
    git clone https://github.com/kolithawarnakulasooriya/Docker-Workshop.git
    b. Otherwise, you can download the project as a .zip file using “Donwload Zip” option

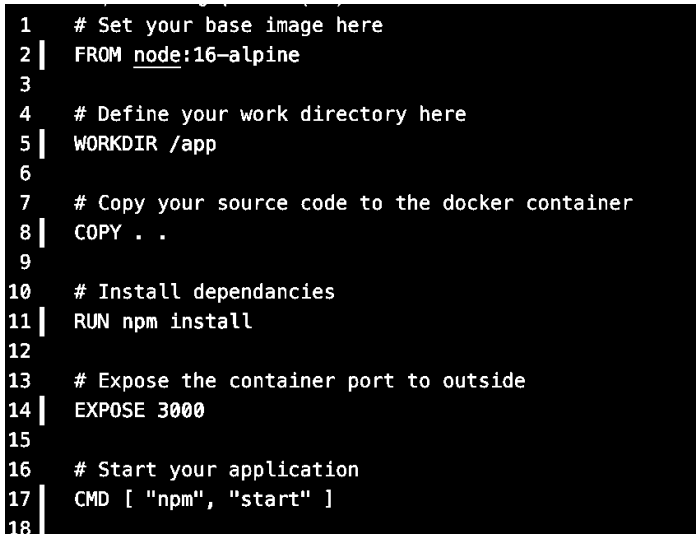


4. Create a docker hub account with your email

    https://www.docker.com/products/docker-hub/

Configure your Docker-Workshop project

1. Open your “Docker-Workshop” project from Visual Studio Code Editor.
    a. **Method 1:**
        i. Open Visual studio code
        ii. Click File -> Open Folder. And select your project folder
    b. **Method 2:**
        i. Open Terminal
        ii. Go to your project directory
        cd <path to directory>/ Docker-Workshop
        iii. Open the project using the code editor
        code .
2. Open Docker File from the editor window
3. Configure your base image. We are using “node 16 alpine” image
   FROM node:16-alpine
4. Configure the working directory
   WORKDIR /app
5. Copy your source files to the working directory
   COPY . .
6. Install dependencies
   RUN npm install
7. Expose the port from the container to outside
   EXPOSE 3000
8. Run your application
   CMD [“npm”, “start”]
9. Finally, your DockerFile looks like below



```
# Set your base image here
FROM node:16-alpine

# Define your work directory here
WORKDIR /app

# Copy your source code to the docker container
COPY . .

# Install dependancies
RUN npm install

# Expose the container port to outside
EXPOSE 3000

# Start your application
CMD [ "npm", "start" ]
```

Build your Docker image

1. Open your terminal
2. Go to your project directory

   cd <path to directory>/ Docker-Workshop

3. Build your image using the below command
   docker build -t docker-workshop:latest .
   a. -t : this is specifying a tag to the docker image
4. After building the docker image, check your image is in the docker image list

   docker image ls

Run your Docker container

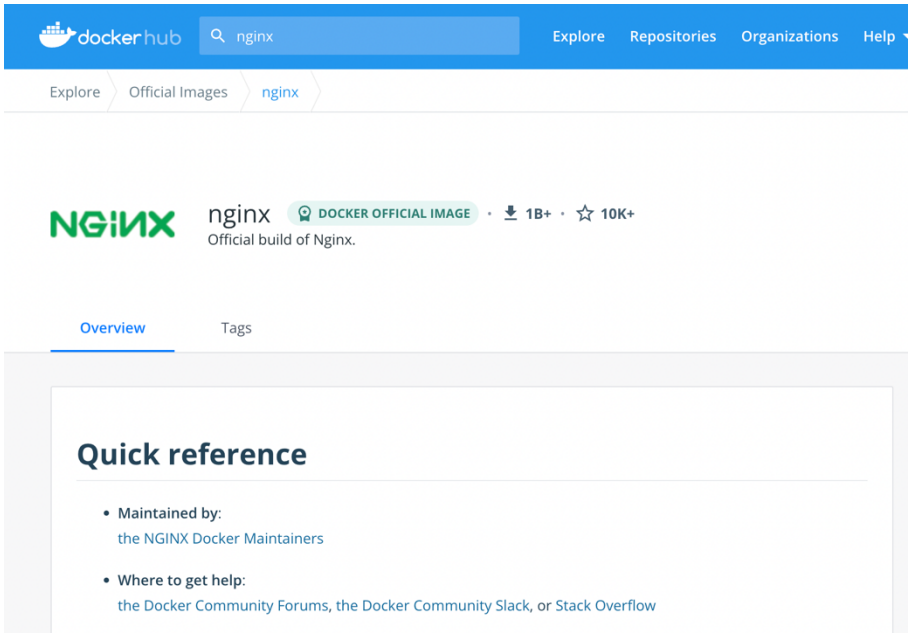
1. Open your terminal
2. Go to your project directory

   cd <path to directory>/ Docker-Workshop

3. Run your build image
   docker run -p 3000:3000 docker-workshop:latest
4. Go to the browser and type http://localhost:3000 to load the docker-workshop web application
5. Run your build image in detached mode
   docker run -d -p 3000:3000 docker-workshop:latest
6. List your containers
   docker ps
7. Copy the container id and Kill the container
   docker kill <container_id>

Get your first image from docker hub

1. Go to https://hub.docker.com/
2. Type “nginx” and get nginx repository or goto https://hub.docker.com/_/nginx

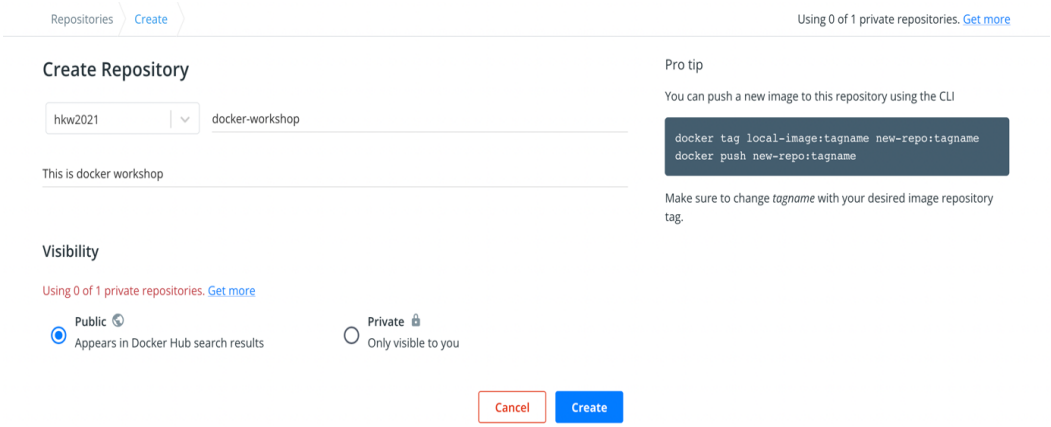


3. Pull the nginx image using the below command
   `docker pull nginx`
4. Run the nginx container
   `docker run -p 3000:80 nginx:latest`
5. Go to the browser and type http://localhost:3000 to load the nginx initial page

Share your image with the docker hub

6. Go to https://hub.docker.com/
7. Click on the repositories tab or go to https://hub.docker.com/repositories

8. Create new repository

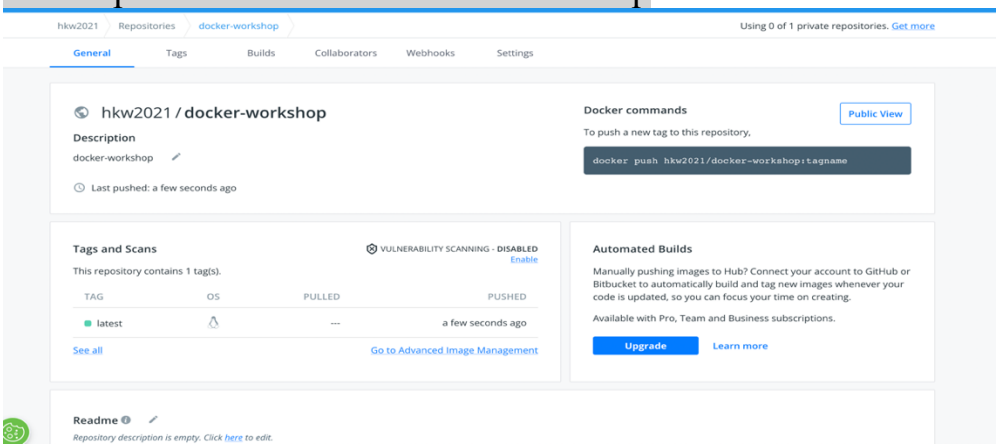


9. Open your terminal
10. Go to your project directory
    cd <path to directory>/ Docker-Workshop
11. Tag your current image as per the new repo
    docker tag docker-workshop:latest <username>/docker-workshop:latest
12. Authenticate your local docker

    docker login

    username: docker hub username
    password: docker hub password

13. Push your docker image to the docker hub
    docker push <username>/docker-workshop



Pull docker image from docker hub

1. Remove all docker images
   docker image rm <image id>
2. Pull the image we pushed to the docker hub
   docker pull <username>/docker-workshop:latest

3. Run your newly pulled image
   `docker run -p 3000:3000 <username>/docker-workshop:latest`
4. Go to the browser and type http://localhost:3000 to load the docker-workshop web application

Useful commands

1. Check docker version
   `docker version`
2. Get information of current platform
   `docker info`
3. List all containers
   `docker ps -a`
4. Remove docker container
   `docker rm <container_id>`
5. Copy files container to host or host to container
   `docker cp <container_id>:<path> <host_path>`
   `docker cp <host_path> <container_id>:<path>`
6. Delete all unused images
   `docker image prune -a`
7. Delete all unused containers
   `docker container prune`
8. Check logs
   `docker logs`
9. List all networks available for docker container
   `docker network ls`
10. List all volumes
    `docker volume ls`