Dockers 101 – Series 7 of N – Setting up a NodeJs Docker Application

PUBLISHED ON *April 5, 2018April 6, 2018 by Mohd Naeem*

- **Requirement**:
  - Setting up a NodeJs Docker Application
- **Strategy**:
  - Create the files needed to run the NodeJS application
  - Create a Dockerfile
  - Build, run , push and pull the image
  - How to use **ONBUILD** to delay a dependency till build time
- **Solution:**
  - Login to your Host machine(in my case a CentOS 7 machine)
  - Make a directory "**mynodejs**" and go to the directory – **mkdir mynodejs && cd mynodejs**
  - Create a file **package.json** with the following component and save

    -
      ```
      {

        "name": "my_docker_nodejs_app",
        "version": "1.0.0",
        "description": "My Docker NodeJs App",
        "author": "MOhd Naeem <naeem.mohd@hotmail.com>",
        "main": "server.js",

        "scripts": {

          "start": "node server.js"

        },

        "dependencies": {

          "express": "^4.16.1"

        }

      }
      ```
  - Create a file **server.js** with the following component and save
    -

```
'use strict';
const express = require('express');
// Constants
const PORT = 8080;
const HOST = '0.0.0.0';
// App
const app = express();
app.get('/', (req, res) => {
res.send('Hello world\n');
});
app.listen(PORT, HOST);
console.log(`Running on http://$ (http://$){HOST}:${PORT}`);
```

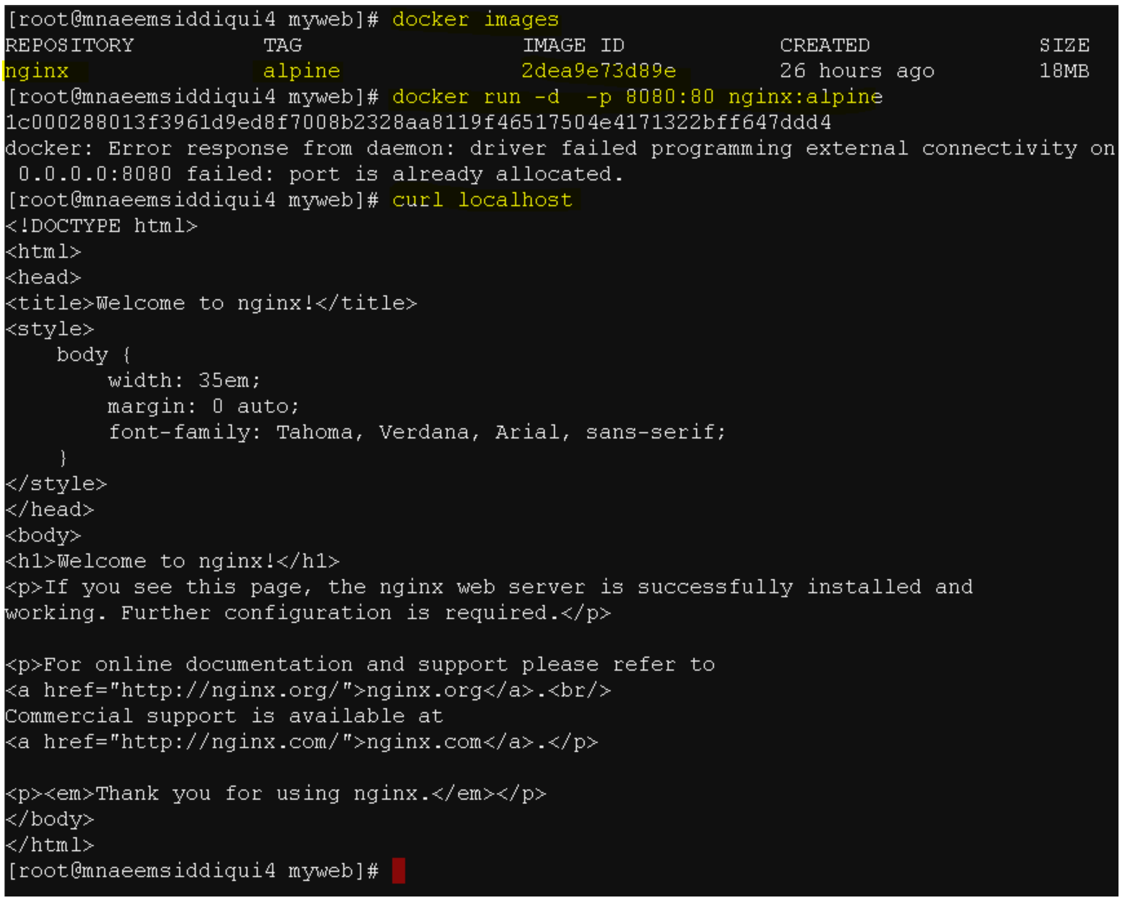
- Create a file **Dockerfile** with the following component and save

  -
    ```
    # starting from base image node:alpine
    FROM node:7-alpine
    # Creating an app directory on the container
    RUN mkdir -p /src/app
    # setup working directory
    WORKDIR /src/app
    # Installing any app dependencies
    # A wildcard being used to ensure both package.json and package-lock.json are copied
    # if nodejs V>5+
    COPY package*.json /src/app
    # For PROD env only use flag –only=production
    # e.g RUN npm install –only=production
    # Running npm install in Non-Prod env.
    RUN npm install
    # Bundle app source
    COPY . /src/app

    # Expose port 3000
    EXPOSE 3000

    # Run command to start npm
    CMD [ "npm", "start" ]
    ```

- Create a file **.dockerignore** with the following component and save

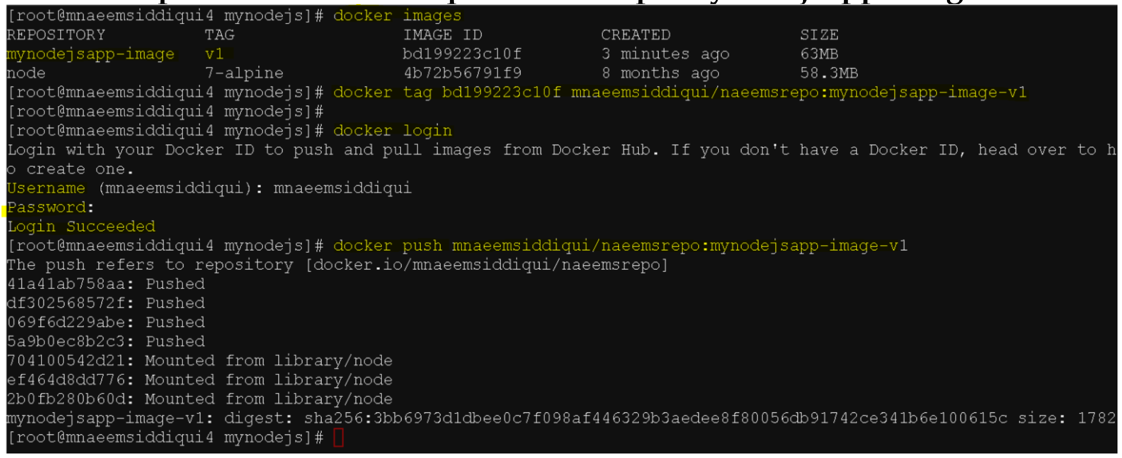
  -
    ```
    node_modules
    npm-debug.log
    ```

- Now build the app-
  - **docker build -t mynodejsapp-image:v1 .**
- Now run run the container to run the website
  - **docker run -d -p 49160:8080 mynodejsapp-image:v1**
- Check the content
  - **curl -i localhost:49160**



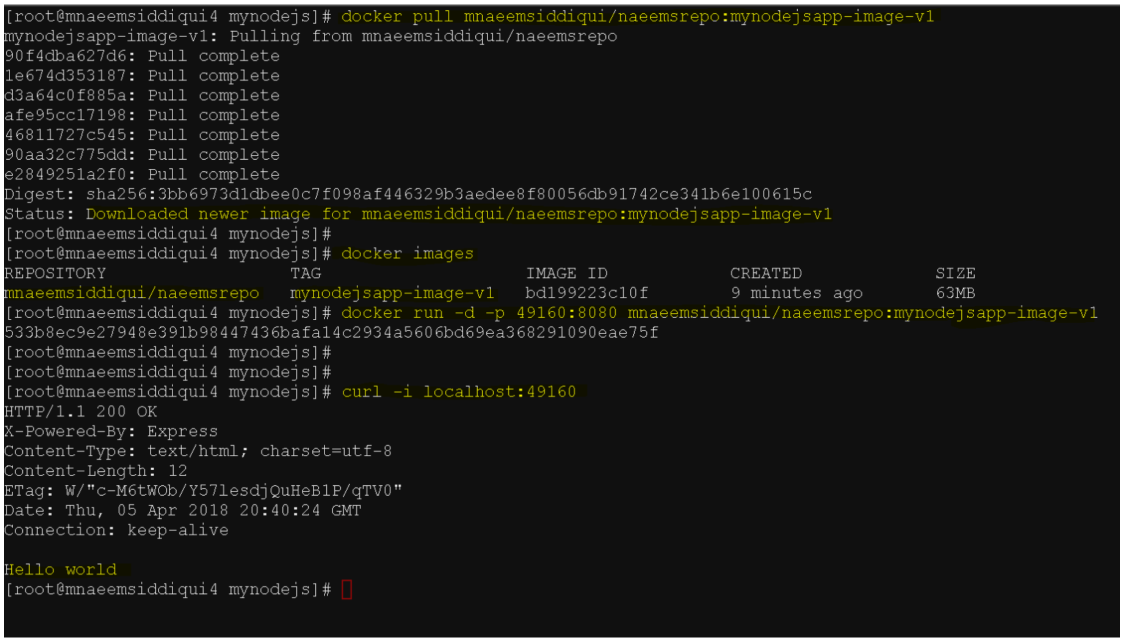
```
[root@mnaeemsiddiqui4 myweb]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               alpine              2dea9e73d89e        26 hours ago        18MB
[root@mnaeemsiddiqui4 myweb]# docker run -d  -p 8080:80 nginx:alpine
1c000288013f3961d9ed8f7008b2328aa8119f46517504e4171322bff647ddd4
docker: Error response from daemon: driver failed programming external connectivity on
 0.0.0.0:8080 failed: port is already allocated.
[root@mnaeemsiddiqui4 myweb]# curl localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@mnaeemsiddiqui4 myweb]#
```

- Now check for the image name for your app and tag it for pushing it to Docker Hub
  - **docker images** # to check for image name
  - **docker tag image username/repository:tag** # for tagging
    - **docker tag 4ffd91cdc6a0 mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1**
  - **docker login** # to login to the Docker hub
- Now push the image to Docker Hub
  - **docker push mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1**



```
[root@mnaeemsiddiqui4 mynodejs]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mynodejsapp-image   v1                  bd199223c10f        3 minutes ago       63MB
node                7-alpine            4b72b56791f9        8 months ago        58.3MB
[root@mnaeemsiddiqui4 mynodejs]# docker tag bd199223c10f mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1
[root@mnaeemsiddiqui4 mynodejs]#
[root@mnaeemsiddiqui4 mynodejs]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to h
o create one.
Username (mnaeemsiddiqui): mnaeemsiddiqui
Password:
Login Succeeded
[root@mnaeemsiddiqui4 mynodejs]# docker push mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1
The push refers to repository [docker.io/mnaeemsiddiqui/naeemsrepo]
41a41ab758aa: Pushed
df302568572f: Pushed
069f6d229abe: Pushed
5a9b0ec8b2c3: Pushed
704100542d21: Mounted from library/node
ef464d8dd776: Mounted from library/node
2b0fb280b60d: Mounted from library/node
mynodejsapp-image-v1: digest: sha256:3bb6973d1dbee0c7f098af446329b3aedee8f80056db91742ce341b6e100615c size: 1782
[root@mnaeemsiddiqui4 mynodejs]#
```

- Now pull the image to Docker Hub
  - **docker pull mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1**
- Now run it on another server
  - **docker run -d -p 49160:8080 mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1**
  - **curl -i localhost:49160**

- 

```
[root@mnaeemsiddiqui4 mynodejs]# docker pull mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1
mynodejsapp-image-v1: Pulling from mnaeemsiddiqui/naeemsrepo
90f4dba627d6: Pull complete
1e674d353187: Pull complete
d3a64c0f885a: Pull complete
afe95cc17198: Pull complete
46811727c545: Pull complete
90aa32c775dd: Pull complete
e2849251a2f0: Pull complete
Digest: sha256:3bb6973d1dbee0c7f098af446329b3aedee8f80056db91742ce341b6e100615c
Status: Downloaded newer image for mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1
[root@mnaeemsiddiqui4 mynodejs]#
[root@mnaeemsiddiqui4 mynodejs]# docker images
REPOSITORY                  TAG                    IMAGE ID            CREATED             SIZE
mnaeemsiddiqui/naeemsrepo   mynodejsapp-image-v1   bd199223c10f        9 minutes ago       63MB
[root@mnaeemsiddiqui4 mynodejs]# docker run -d -p 49160:8080 mnaeemsiddiqui/naeemsrepo:mynodejsapp-image-v1
533b8ec9e27948e391b98447436bafa14c2934a5606bd69ea368291090eae75f
[root@mnaeemsiddiqui4 mynodejs]#
[root@mnaeemsiddiqui4 mynodejs]#
[root@mnaeemsiddiqui4 mynodejs]# curl -i localhost:49160
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 12
ETag: W/"c-M6tWOb/Y57lesdjQuHeB1P/qTV0"
Date: Thu, 05 Apr 2018 20:40:24 GMT
Connection: keep-alive

Hello world
[root@mnaeemsiddiqui4 mynodejs]#
```

- Using OnBuild to delay execution of dependencies
- Lets update the Dockerfile with content below
- The big difference is that we are delaying the execution of commands for copying the package.json, npm install and copying of source application files till building by using keyword **build**
-

  -
    ```
    #starting from base image node:alpine
    FROM node:7-alpine
    # Creating an app directory on the container
    RUN mkdir -p /src/app
    # setup working directory
    WORKDIR /src/app
    # Installing any app dependencies
    # A wildcard being used to ensure both package.json and package-lock.json are copied
    # if nodejs V>5+
    ONBUILD COPY package*.json /src/app
    # For PROD env only use flag –only=production
    # e.g RUN npm install –only=production
    # Running npm install in Non-Prod env.
    ONBUILD RUN npm install
    # Bundle app source
    ONBUILD COPY . /src/app

    # Expose port 3000
    EXPOSE 3000

    # Run command to start npm
    CMD [ “npm”, “start” ]
    ```
- Now build and run the application once again.
-

CATEGORIES DOCKERS