
Running NodeJs App inside Docker Container

Ganesh Pol

16-Nov-2022

Introduction	1
Project Setup	1
Important project files	1
Setting up a project and Running it	2
Installing node js	2
Install required dependencies for the project	2
Packaging application as a docker image	3
Dockerfile details	3
Create .dockerignore file	3
Create docker image	3
Docker run	4

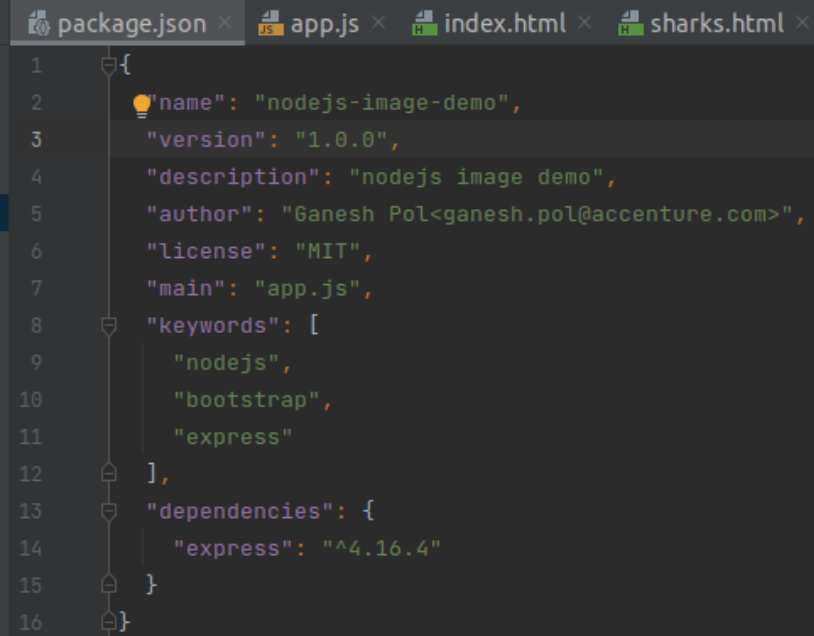
Introduction

- It is a simple example of how to run a node js project in docker.
- Running the application as a docker container provides the following benefits.
 - Minimize impedance mismatch between local development machine and production machine.
 - The same application can be deployed in
 - AWS Lambda
 - EKS
 - ECS
 - Or on EC2 as well.

Project Setup

Important project files

Node js project on a high level consists of package.json. It essentially consists of package-level dependencies



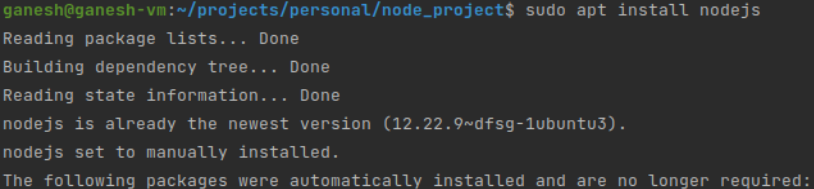
```

1  {
2    "name": "nodejs-image-demo",
3    "version": "1.0.0",
4    "description": "nodejs image demo",
5    "author": "Ganesh Pol<ganesh.pol@accenture.com>",
6    "license": "MIT",
7    "main": "app.js",
8    "keywords": [
9      "nodejs",
10     "bootstrap",
11     "express"
12   ],
13   "dependencies": {
14     "express": "^4.16.4"
15   }
16 }
```

Just by looking at package.json, we can see that it has dependency on node js package express with version 4.16.4

Setting up a project and Running it

Installing node js



```

ganesh@ganesh-vm:~/projects/personal/node_project$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nodejs is already the newest version (12.22.9~dfsg-1ubuntu3).
nodejs set to manually installed.
The following packages were automatically installed and are no longer required:
```

```
ganesh@ganesh-vm:~/projects/personal/node_project$ node -v
v12.22.9
```

```
ganesh@ganesh-vm:~/projects/personal/node_project$ sudo apt install npm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
npm is already the newest version (8.5.1~ds-1).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver intel-media-
  libcodec2-1.0 libdav1d5 libflashrom1 libflite1 libftdi1-2 libgme0 libgsm1
  librubberband2 libserd-0-0 libshine3 libsnappy1v5 libsord-0-0 libsratom-0-
  libvidstab1.1 libx265-199 libxvidcore4 libzim2 libzmq5 libzvbi-common lib
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
```

Install required dependencies for the project

- Npm list -g will provide a list of global packages.

```
ganesh@ganesh-vm:~/projects/personal/node_project$ npm list -g
/usr/local/lib
└─ (empty)
```

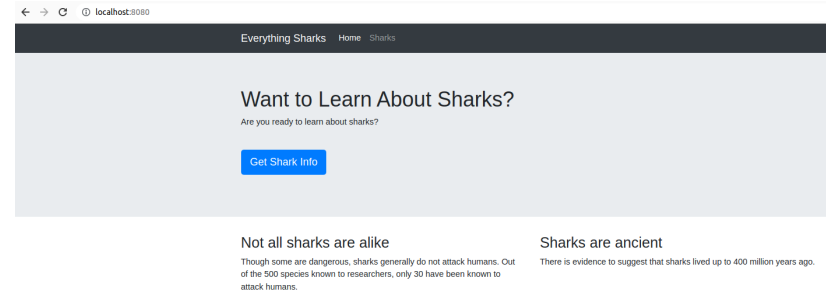
```
ganesh@ganesh-vm:~/projects/personal/node_project$ npm install
added 57 packages, and audited 58 packages in 13s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Running project locally

```
ganesh@ganesh-vm:~/projects/personal/node_project$ node app.js
Example app listening on port 8080!
```



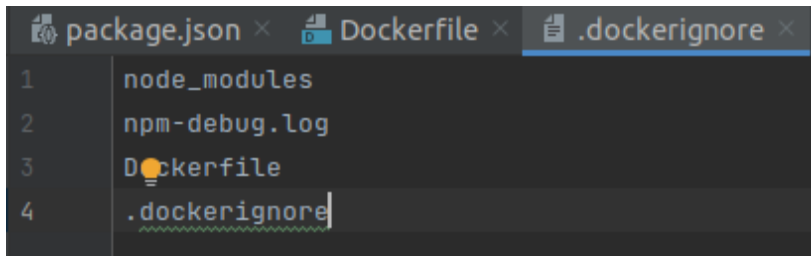
Packaging application as a docker image

Dockerfile details

```
# This image includes node js and npm.
FROM node:12.22.9-alpine3.15
# create node_modules sub directory in /home/node/app
# change ownership to node user
RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app
# set work dir
WORKDIR /home/node/app
# copy package*.json. it will copy both package and package-lock
COPY package*.json ./
# ensure that all application files are owned by the non root [node] user
#including content of node_modules dir ,switch the user to node
USER node
# install dependencies
RUN npm install
# copy application code with the appropriate permission to app dir on container
COPY --chown=node:node . .
# expose port 8080 on the container and start application.
EXPOSE 8080
CMD [ "node", "app.js" ]
```

Create .dockerignore file

It will ignore these files when docker try to copy.



```
package.json x Dockerfile x .dockerignore x
1 node_modules
2 npm-debug.log
3 Dockerfile
4 .dockerignore
```

Create docker image

```
docker build -t nodejs-example-demo .
```

```
ganesh@ganesh-vm:~/projects/personal/node_project$ docker build -t nodejs-example-demo .
Sending build context to Docker daemon 56.32kB
Step 1/9 : FROM node:12.22.9-alpine3.15
12.22.9-alpine3.15: Pulling from library/node
59bf1c3509f3: Pull complete
56ea1bcfc516: Pull complete
55284a5c72ce: Pull complete
a9867c49b9eb: Pull complete
Digest: sha256:c3e7817c8fd124f1597cd66124d247db8e138e8ef311ba085b7104d900129d0b
```

```
ganesh@ganesh-vm:~/projects/personal/node_project$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
nodejs-example-demo latest      328673a36f6b 9 seconds ago 94MB
```

Docker run

```
docker run --name my-node-js-app -p 81:8080 -d
nodejs-example-demo:latest
```

```
ganesh@ganesh-vm:~/projects/personal/node_project$ docker run --name my-node-js-app -p 81:8080 -d nodejs-example-demo:latest
323fc92610d74a6189d8393723b13688f099fbd40cd547c0b94b3fd67fa074
ganesh@ganesh-vm:~/projects/personal/node_project$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                    NAMES
323fc92610d   nodejs-example-demo:latest  "docker-entrypoint.s..." 18 seconds ago Up 15 seconds 0.0.0.0:81->8080/tcp, :::81->8080/tcp  my-node-js-app
```

