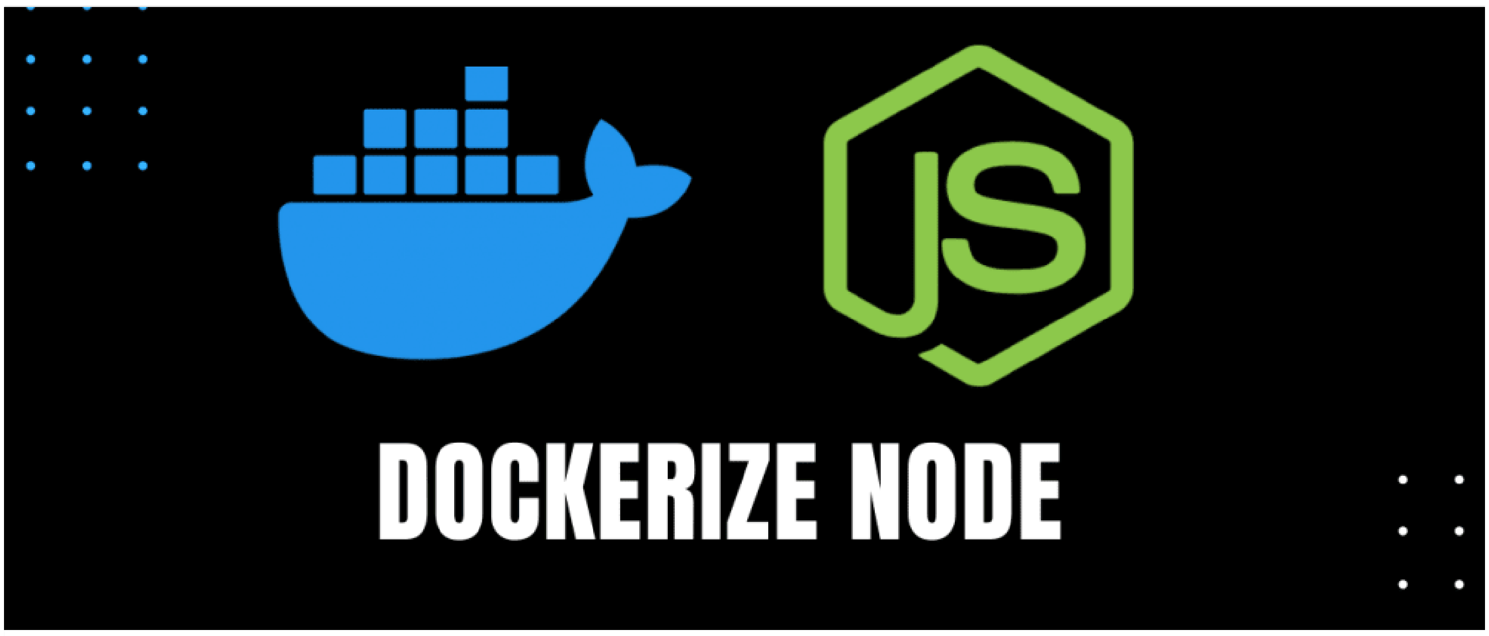


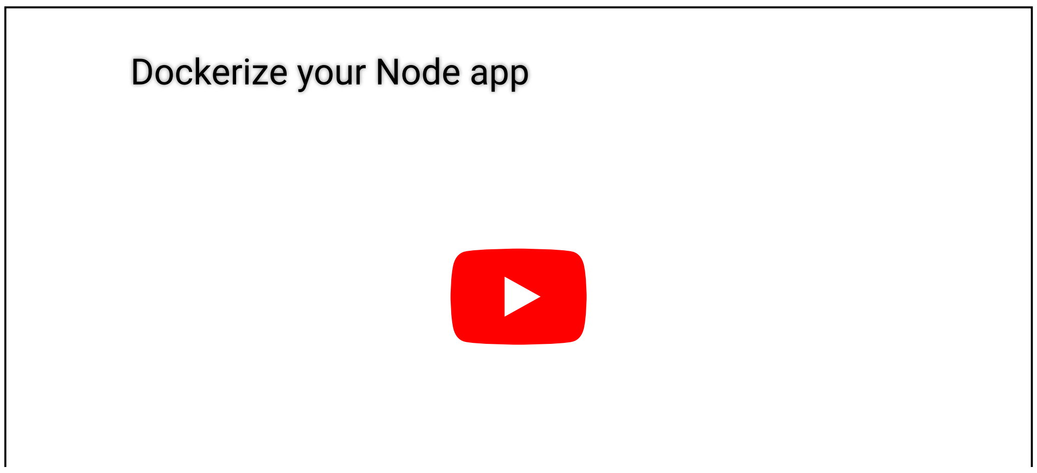
Karan Pratap Singh
Posted on 13 Jul 2021

Dockerize your Node app

#node #docker #devops #javascript

Hey, welcome back. This article is part of the Dockerize series, make sure to checkout the Introduction where I go over some concepts we are going to use.

Today we'll dockerize our Node application, very similar to how we dockerized our React app in the last part by taking advantage of builder pattern with multi stage builds!



I've also made a video, if you'd like to follow along

Project setup

I've initialized a simple express app

```
├── node_modules
├── index.js
├── package.json
└── yarn.lock
```

```
const express = require('express');

const app = express();
const PORT = process.env.PORT || 4000;

app.get('/', (request, response) => {
  response.status(200).json({
    message: 'Hello Docker!',
  });
});

app.listen(PORT, () => {
  console.log(`Server is up on localhost:${PORT}`);
});
```

I've also setup esbuild to bundle our project.

```
"build": "esbuild --bundle src/index.js --outfile=build/app.js --minify --platform=node"
```

For more details, you can checkout my previous article *Blazing fast TypeScript with Webpack and ESBuild*.

For development

Let's start by adding a `Dockerfile`

```
FROM node:14-alpine AS development
ENV NODE_ENV development
# Add a work directory
WORKDIR /app
# Cache and Install dependencies
COPY package.json .
COPY yarn.lock .
RUN yarn install
# Copy app files
COPY . .
# Expose port
```

```
EXPOSE 4000

# Start the app
CMD [ "yarn", "start" ]
```

Let's create a `docker-compose.dev.yml`. Here we'll also mount our code in a volume so that we can sync our changes with the container while developing.

```
version: "3.8"

services:
  app:
    container_name: app-dev
    image: app-dev
    build:
      context: .
      target: development
    volumes:
      - ./src:/app/src
    ports:
      - 4000:4000
```

Let's update our `package.json` scripts

```
"dev": "docker-compose -f docker-compose.dev.yml up"
```

we can use the `-d` flag to run in daemon mode

Let's start developing!

```
yarn dev
```

Great, our dev server is up!

```
Attaching to app-dev
app-dev  | yarn run v1.22.5
app-dev  | $ nodemon src/index.js
app-dev  | [nodemon] to restart at any time, enter `rs`
app-dev  | [nodemon] watching path(s): *.*
app-dev  | [nodemon] starting `node src/index.js`
app-dev  | Server is up on localhost:4000
```

For production

```
FROM node:14-alpine AS builder
ENV NODE_ENV production
# Add a work directory
WORKDIR /app
# Cache and Install dependencies
COPY package.json .
COPY yarn.lock .
RUN yarn install --production
# Copy app files
COPY . .
# Build
```

```
CMD yarn build


FROM node:14-alpine AS production
# Copy built assets/bundle from the builder
COPY --from=builder /app/build .
EXPOSE 80
# Start the app
CMD node app.js
```

Let's add a `docker-compose.prod.yml` for production

```
version: "3.8"

services:
  app:
    container_name: app-prod
    image: app-prod
    build:
      context: .
      target: production
```

```
docker-compose -f docker-compose.prod.yml build
```

let's start our production container on port `80` with the name `react-app`

```
docker run -p 80:4000 --name node-app app-prod
```

Next steps

With that, we should be able to take advantage of docker in our workflow and deploy our production images faster to any platform of our choice.

Feel free to reach out to me on Twitter if you face any issues.

Dockerize series (5 Part Series)

Discussion (4)

Jon Lauridsen • Jul 14 '21 ...

Thanks! Where do you run tests?

Karan Pratap Singh • Jul 15 '21

Thank you reading! I usually run integration tests before building the images and unit test while building by adding a step in the dockerfile itself

Allan Camilo • Jul 13 '21

great article

Karan Pratap Singh • Jul 16 '21

Thank you Allan!

Code of Conduct • Report abuse

🤔 **Did you know?**

🗺 We put together some guides that include some of our favorite content on DEV. Check them out here.

Karan Pratap Singh

A software engineer who values learning and growing with people, teams, and technologies.

WORK
Senior Software Engineer

JOINED
15 Mar 2020

More from Karan Pratap Singh

Top 5 Docker Best Practices
#docker #tutorial #beginners

Connecting to PostgreSQL using GORM
#go #postgres #docker

Preview Environments with AWS & Cloudflare
#aws #typescript #docker #devops