# How To Dockerize a Node.js Application?

By Naveen PS

MAKERDEMY
LEARN EXPLORE MAKE

# How to dockerize a Node.js Application

## Install Node.js

Install NPM Manager : nodejs.org/en/download/package-manager

## Install NPM and Express Framework

In addition to npm, our application will use the Express Framework, one of the most popular Node.js frameworks. Create a new directory and initialize npm:

```
$ mkdir helloworld
$ cd helloworld
$ npm init
```

When asked for the details of the application just confirm the default values. The package`.json` will hold the dependencies of the app. Now lets add the Express Framework as the first dependency:

```
$ npm install express --save
```

The file will look like this

```
{
  "name": "helloworld",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.15.2"
  }
}
```

## Hello World

With everything installed, we can create an `index.js` file with a simple HTTP server that will serve our Hello World website:

```
//Load express module with `require` directive
var express = require('express')
var app = express()

//Define request response in root URL (/)
app.get('/', function (req, res) {
  res.send('Hello World!')
})

//Launch listening server on port 8081
app.listen(8081, function () {
  console.log('app listening on port 8081!')
})
```

## Run the app

The application is ready to launch:

```
$ node index.js
```

Go to `http://localhost:8081/` in your browser to view it.

## Install Docker

Begin with installing Docker for your type of OS:

- MacOS
- Windows
- Ubuntu
- Debian

## Write Dockerfile

The Docker container is launched on the basis of a Docker image, a template with the application details. The Docker image is created with instructions written in the Dockerfile. Let's add `Dockerfile` to the directory with our application:
**Line 1**: Use another Docker image for the template of my image. We shall use the official Node.js image with Node v7.

```
FROM node:7
```

**Line 2**: Set working dir in the container to `/app`. We shall use this directory to store files, run npm, and launch our application:

```
WORKDIR /app
```

**Line 3-5**: Copy application to `/app` directory and install dependencies. If you add the `package.json` first and run `npm install` later, Docker won't have to install the dependencies again if you change the `package.json` file. This results from the way the Docker image is being built (layers and cache), and this is what we should do:

```
COPY package.json /app
RUN npm install
COPY . /app
```

**Line 6**: This line describes what should be executed when the Docker image is launching. What we want to do is to run our application:

```
CMD node index.js
```

**Line 7**: Expose port 8081 to the outside once the container has launched:

```
EXPOSE 8081
```

Summing up, the whole `Dockerfile` should look like this:

```
FROM node:7
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app
CMD node index.js
EXPOSE 8081
```

## Build Docker image

With the instructions ready all that remains is to run the `docker build` command, set the name of our image with `-t` parameter, and choose the directory with the Dockerfile:

```
$ docker build -t hello-world .
```

## Run Docker container

The application has been baked into the image. Dinner time! Execute the following string to launch the container and publish it on the host with the same port 8081:

```
docker run -p 8081:8081 hello-world
```

# THE END