# Packaging, deploying, and running a Node application using Docker

## Objective:

In this exercise, you will learn how to create a docker for a Node.js application.   
Docker is a powerful tool that enables developers to package their applications and dependencies into a container, making it easier to deploy, manage, and scale their applications. Although the following is a simple example, the same procedure can be applied to a more complex project.

## Prerequisites

Before starting the exercise, you should install Node.js, NPM and Docker packages installed on your local PC.

## Steps by step instructions

* Create a Node.js application:

Create a simple Node.js application that prints "Hello World" to the console.

* You will then create a Node app which serves some content.
* You will also create a file called **package.json** (using the **npm init** command) which defines all the project dependencies and sets other settings.   
  You will also experiment with installing a common module called “express”..

For this exercise, we will create a new directory called **docker\_app** and create a file called **server.js** inside it which holds a few simple lines of JavaScript for running a Node application.

mkdir docker\_app

cd docker\_app

npm init -y

code .

* The contents of server.js will be:

'use strict';

const express = require('express');

// Constants

const PORT = 9000;

const HOST = '0.0.0.0';

// App

const app = express();

app.get('/', (req, res) => {

  res.send('<h1>Hello World!</h1>');

});

app.listen(PORT, HOST, () => {

  console.log(`Running on http://${HOST}:${PORT}`);

});

* Change the contents of **package.json** to set the dependencies and other attributes.

{

  "name": "qa\_web\_app",

  "version": "1.0.0",

  "description": "Node.js using Docker",

  "author": "mike <mike@qa.com>",

  "main": "server.js",

  "scripts": {

    "start": "node server.js"

  },

  "dependencies": {

    "express": "^4.16.1"

  }

}

* Change the author name to you name.
* Install all the dependencies by typin the following command in the Terminal window  
  **npm install**
* Start running you node app to see if it works by typing: **npm start**
* Open a browser and type: <http://localhost:9000>

You should see Graphical user interface, text, application, chat or text message

Description automatically generated

* You can now stop the app running by pressing Ctrl-C in the Terminal window.

Type y to the question “Terminate batch job (Y/N)? ”

## **Create a Docker container for your app**

### **Create a Dockerfile:**

In this part, you will create a Dockerfile that will be used to build a Docker image of your Node.js application.

* Create a file called **Dockerfile** in the same folder as server.js
* Include the following instructions for selecting the node version 16 and installing the files on the client.

FROM node:16

# Create app directory

WORKDIR /usr/src/app

# Install app dependencies

# A wildcard is used to ensure both package.json AND package-lock.json are copied

# where available (npm@5+)

COPY package\*.json ./

RUN npm install

# If you are building your code for production

# RUN npm ci --only=production

# Bundle app source

COPY . .

EXPOSE 9000

CMD [ "node", "server.js" ]

* Create a **.dockerignore** (dot dockerignore) file in the same directory as your Dockerfile with following content:

node\_modules

npm-debug.log

We will not copy the module files into the image because they can be created (npm install).

### **Build the Docker image**

* Open a terminal window, navigate to the root directory of your Node.js application
* Run the following command to build the Docker image with the tag of say **mike/qa-web-app**   
  **docker build . -t mike/qa-web-app**

Please note the dot ‘.’ In the middle of the command.  
You can change the tag but please make sure each letter of the tag is in lowercaseThis command will use the Dockerfile to build a Docker image of your Node.js application and tag it with the name that you soecify.

### Run the Docker container:

* After the Docker image has been built, you can run a Docker container using the following command: **docker run -p 5000:9000 mike/qa-web-app**

This command will start a new Docker container based on the node-docker-app image and map port **9000 in the container** to port **5000 on your local machine** (the host).

So the format is **-p <host port>:<container port>**.

Test the Docker container:

* You should be able to access your Node.js application by opening a web browser and navigating to **http://localhost:5000**.

## Now let’s see how to stop your docker

* Close the browser where Hello World was displayed and then type the following command in the Terminal window: **docker container ls**
* find the image ID of the docker you want to kill and then issue the following command  
  docker kill <id of the container ID you want to kill>

Congratulations, you have successfully created a Docker to host a Node application.