

## Module 10: Microservices Application

---

### Demo Document 1

edureka!

**edureka!**

© Brain4ce Education Solutions Pvt. Ltd.

## Microservices with Docker

### DEMO Steps:

**Step 1:** First create simple node app to deploy over docker. For that first we have created package.json file

```
Avyaans-MacBook-Pro:microservice avi$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (microservice) microservice
version: (1.0.0)
description: microservice with node and docker
entry point: (index.js)
test command:
git repository:
keywords: node docker
author: edureka
license: (ISC)
About to write to /Users/avi/Desktop/folder/EdurekaApp/module10/microservice/package.json:
{
  "name": "microservice",
  "version": "1.0.0",
  "description": "microservice with node and docker",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "node",
    "docker"
  ],
  "author": "edureka",
  "license": "ISC"
}
Is this OK? (yes) █
```

**Step 2:** Now install axios for calling api and express as nodejs framework for routing in the application

```
Avyaans-MacBook-Pro:microservice avi$ npm i axios express
```

**Step 3:** In app.js we will create a basic node server running on the port 8000 using express server

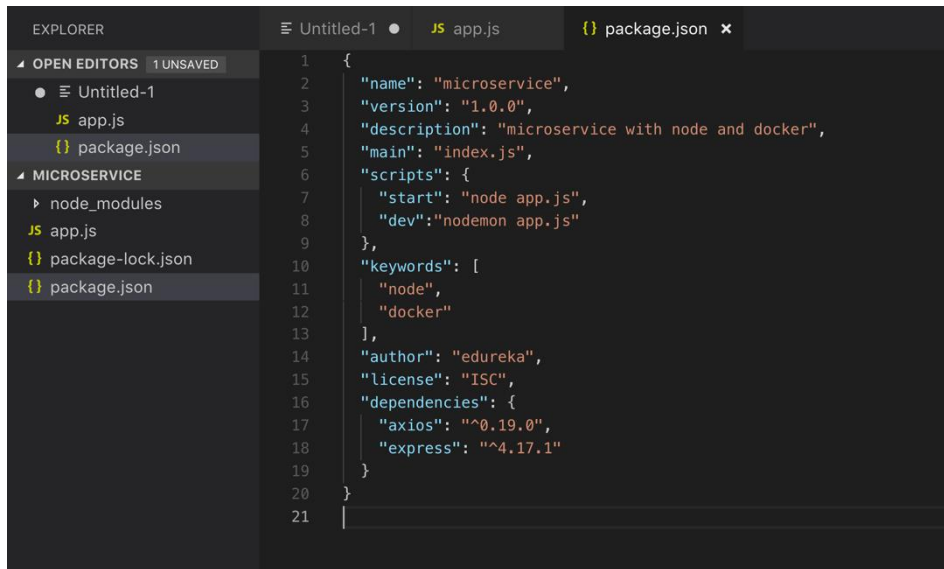
```
1 const express = require('express');
2 const axios = require('axios');
3
4 // Constants
5 const PORT = 8080;
6 const HOST = '0.0.0.0';
7
8 app.listen(PORT, HOST);
9 console.log(`Running on http://${HOST}:${PORT}`);
```

**Step 4:** To call the api we are using axios. Create one **get** end point as a default route and response using **res.send**.

```
1 const express = require('express');
2 const axios = require('axios');
3
4 // Constants
5 const PORT = 8080;
6 const HOST = '0.0.0.0';
7
8 const app = express();
9 const searchUrl = 'https://restcountries.eu/rest/v2/all';
10
11 app.get('/', (req, res) => {
12   axios.get(searchUrl).then(response => {
13     const responseJSON = response.data;
14     return res.status(200).json({ source: 'Docker Microservice', ...responseJSON, });
15   })
16   .catch(err => {
17     return res.json(err);
18   });
19 }
20 );
21
22 app.listen(PORT, HOST);
23 console.log(`Running on http://${HOST}:${PORT}`);
```

**Step 5:** To use the application we need to make both dev and start command.

In **dev** use nodemon and in **start** use app.js to execute the application.



```
1 {
2   "name": "microservice",
3   "version": "1.0.0",
4   "description": "microservice with node and docker",
5   "main": "index.js",
6   "scripts": {
7     "start": "node app.js",
8     "dev": "nodemon app.js"
9   },
10  "keywords": [
11    "node",
12    "docker"
13  ],
14  "author": "edureka",
15  "license": "ISC",
16  "dependencies": {
17    "axios": "^0.19.0",
18    "express": "^4.17.1"
19  }
20 }
21
```

**Step 6:** On running the end point in browser we are getting the api response as json format and now after this we will start deploying the application over docker and make it as microservice



```
1 // 20190702025817
2 // http://localhost:8083/
3
4 {
5   "0": {
6     "name": "Afghanistan",
7     "topLevelDomain": [
8       ".af"
9     ],
10    "alpha2Code": "AF",
11    "alpha3Code": "AFG",
12    "callingCodes": [
13      "93"
14    ],
15    "capital": "Kabul",
16    "altSpellings": [
17      "AF",
18      "Afgānistān"
19    ],
20    "region": "Asia",
21    "subregion": "Southern Asia",
22    "population": 27657145,
23    "latlng": [
24      33,
25      65
26    ],
27    "demonym": "Afghan",
28    "area": 652230,
29    "gini": 27.8,
30    "timezones": [
31      "UTC+04:30"
32    ],
33    "borders": [
34      "IRN",
35
```

**Step 7:** Now to use docker, we need to download and install from the official website according to your OS.

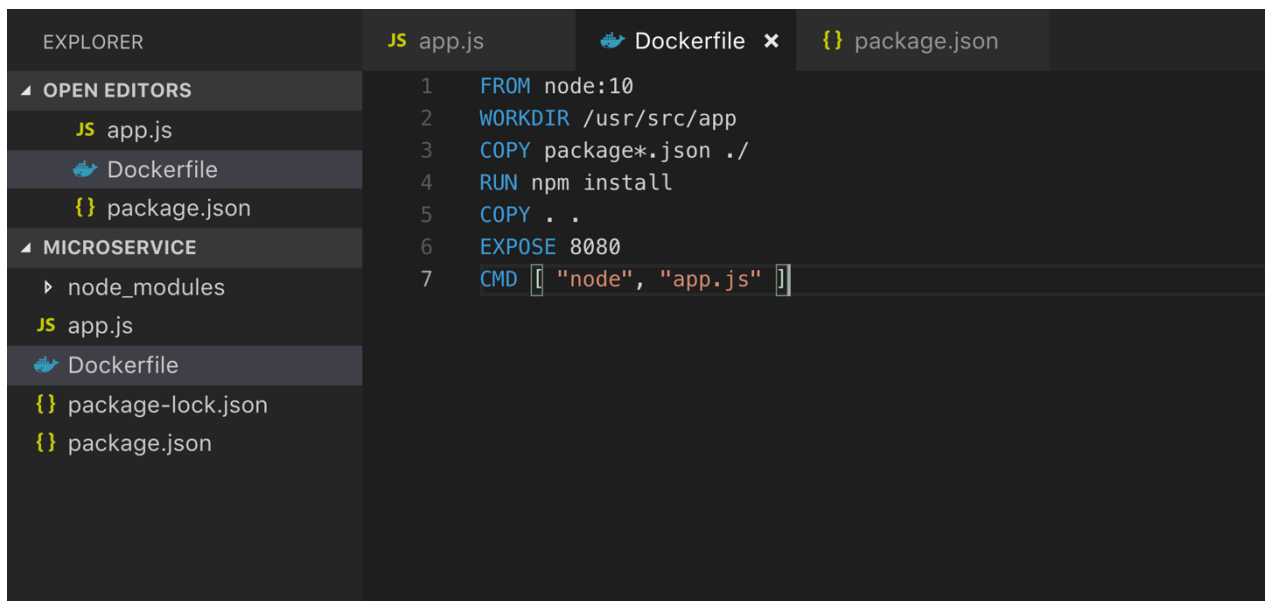


**Step 8:** Now after installation run docker over the machine. After running you can see one docker icon in your application tray. On click on icon you can see Docker is running.



**Step 9:** Now create a docker file in our app folder with following commands

- First install node in docker
- Second create working directory in where you will keep your app
- Now Copy package.json in the working directory
- Write command to install all packages in docker
- After that copy all content in the workdir
- Finally expose the port and excute the command to run the app

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with folders 'OPEN EDITORS' and 'MICROSERVICE'. Under 'OPEN EDITORS', there are files 'app.js', 'Dockerfile', and 'package.json'. Under 'MICROSERVICE', there are subfolders 'node\_modules' and 'app.js', and files 'Dockerfile', 'package-lock.json', and 'package.json'. The main editor area shows the content of the 'Dockerfile' with the following lines:

```
1 FROM node:10
2 WORKDIR /usr/src/app
3 COPY package*.json ./
4 RUN npm install
5 COPY . .
6 EXPOSE 8080
7 CMD ["node", "app.js"]
```

**Step 10:** Once docker file is created, build the image of app using docker command in your console, where -t is the name of the image

```
Avyaans-MacBook-Pro:microservice avi$ docker build . -t ahanda205/countrylist
```

**Step 11:** Create the docker image

## Module 10: Microservices Application

```
Avyaans-MacBook-Pro:microservice avi$ docker build . -t ahanda205/countrylist
Sending build context to Docker daemon 2.686MB
Step 1/7 : FROM node:10
----> e05cbde47b8f
Step 2/7 : WORKDIR /usr/src/app
----> Using cache
----> 83badefd21b6
Step 3/7 : COPY package*.json ./
----> 5918274ceec7
Step 4/7 : RUN npm install
----> Running in c0679ea5ad41
npm WARN microservice@1.0.0 No repository field.

added 58 packages from 41 contributors and audited 131 packages in 1.771s
found 0 vulnerabilities

Removing intermediate container c0679ea5ad41
----> e280fd2b85f5
Step 5/7 : COPY . .
----> 28ef3bbeb0d9
Step 6/7 : EXPOSE 8080
----> Running in 2beb31e8e3ba
Removing intermediate container 2beb31e8e3ba
----> ea231103aab9
Step 7/7 : CMD [ "node", "app.js" ]
----> Running in eb14055b98e4
Removing intermediate container eb14055b98e4
----> bdfa119be585
Successfully built bdfa119be585
Successfully tagged ahanda205/countrylist:latest
Avyaans-MacBook-Pro:microservice avi$
```

**Step 12:** Thus it will generate docker image, that can be excuted to run node application as microservice.

```
Avyaans-MacBook-Pro:microservice avi$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ahanda205/countrylist	latest	bdfa119be585	29 seconds ago	909MB
ahanda205/nodeapp	latest	bd86b3601fc7	18 minutes ago	908MB
<none>	<none>	f85e842aeedc	33 minutes ago	908MB
mreactapp	latest	f282b2878f51	3 days ago	427MB
reactimage	latest	0479fe3ba22e	8 days ago	421MB
<none>	<none>	f878a5676ca7	8 days ago	421MB
<none>	<none>	a08f4759ad0e	8 days ago	421MB
ahanda205/reactprop	tagname	8ec513692a85	9 days ago	423MB
reactprop	latest	8ec513692a85	9 days ago	423MB
<none>	<none>	fab1c1e3c943c	9 days ago	423MB
myreactapp	latest	80cf7f790336	9 days ago	421MB
reactapp	latest	c81204578b01	9 days ago	420MB
node	10	e05cbde47b8f	2 weeks ago	904MB
node	alpine	d4edda39fb81	3 weeks ago	77.8MB

```
Avyaans-MacBook-Pro:microservice avi$
```

**Step 13:** To run the image we need to specify port number on which you want to run the app and the name of the docker image

```
Avyaans-MacBook-Pro:microservice avi$ docker run -p 49163:8080 -d ahanda205/countrylist
37cc7ce8b282b3c0b302f20c0df07e1fc5f60e24724b8cec68a738416cd94bec
Avyaans-MacBook-Pro:microservice avi$
```

**Step 14:** After running the command, we can see that docker container is created, by using command `docker ps` see the created container

```
Avyaans-MacBook-Pro:microservice avi$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
37cc7ce8b282	ahanda205/countrylist	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:49163->8080/tcp	dreamy_volhard
c36b9d3800fc	ahanda205/countrylist	"docker-entrypoint.s..."	2 minutes ago	Up 2 minutes	0.0.0.0:49162->8080/tcp	stupefied_poitras
5ef9ddaca3a3	ahanda205/nodeapp	"docker-entrypoint.s..."	21 minutes ago	Up 21 minutes	0.0.0.0:49161->8080/tcp	nifty_hodgkin
1be7d8fdb54a	f85e842aeedc	"docker-entrypoint.s..."	30 minutes ago	Up 30 minutes	0.0.0.0:49160->8080/tcp	hopeful_bose

```
Avyaans-MacBook-Pro:microservice avi$
```

**Step 15:** If we check the log using docker container id we can see that application is running on docker.

```
[Avyaans-MacBook-Pro:microservice avi$ docker logs 37cc7ce8b282]
Running on http://0.0.0.0:8080
Avyaans-MacBook-Pro:microservice avi$
```

**Step 16:** Finally, when we run app using docker port number we can see the output

This indicate app is running on docker.



The screenshot shows a web browser window with the address bar set to `localhost:49163`. The page displays a JSON object representing data for Afghanistan. The JSON structure is as follows:

```
{
  "0": {
    "name": "Afghanistan",
    "topLevelDomain": [
      ".af"
    ],
    "alpha2Code": "AF",
    "alpha3Code": "AFG",
    "callingCodes": [
      "93"
    ],
    "capital": "Kabul",
    "altSpellings": [
      "AF",
      "Afgānistān"
    ],
    "region": "Asia",
    "subregion": "Southern Asia",
    "population": 27657145,
    "latlng": [
      33,
      65
    ],
    "demonym": "Afghan",
    "area": 652230,
    "gini": 27.8,
    "timezones": [
      "UTC+04:30"
    ],
    "borders": [
      "IRN",

```

### Conclusion:

Thus we have successfully deployed microservices in Docker.