Getting started with Node and express using Docker labs

Introduction

For writing web applications we need server middleware that can run the business logic and expose the same as web services. Webservices are backbone of any modern web application. Modern webservices are lightweight webservices that follow REST paradigm as against legacy SOAP webservices which are heavy. The REST webservices make use of the HTTP protocol aspects like HTTP Verbs and HTTP Headers. Further the modern REST webservices use light weight JSON data format as against heavier XML data format used by legacy SOAP webservices. Thus REST webservices are lightweight and easier to implement than SOAP webservices.

Node.js is a server side programming platform based on Javascript for creating webservices. It uses Google V8 Javascript engine to run the javascript code. Compared to Java based server middleware like Tomcat, Spring Boot et-cetera, the Node.js is a much more lightweight platform. Further developing web services in Node.js platform is much more simpler than devoloping in complex Java based platforms.

Express.js is a framework that further simplifies the programming in Node.js. By using Express.js we can easily develop web services. In this tutorial we shall proceed to familiarize with Node.js itself using the docker shell. Finally we shall proceed to creating a simple Node.js + Express.js service.

Task 1: Play with Node.js in a docker

1. Go to Docker labs https://labs.play-with-docker.com

2. Login using Docker credentials

3. In labs home page, Open a new docker instance by clicking the "add new instance" hyperlink.

4. In the main pane a docker instance shell shall be opened.

5. In the shell type the following command to run a Node.js image in ineteractive mode.
   `docker run -it node:10-slim node`

For full coomand line reference of docker command, check
https://docs.docker.com/engine/reference/commandline/run/

6. Once the node based docker image initializes, an interactive node shell shall be presented.

7. In the node shell one can play with various Javascript expressions.

   - 1 + 2
     (Prints 3)

   - message = 'Hello Word'
     console.log(message)
     (Prints "Hello World")

Task 2: Create a simple Hello World service

1. Follow the first 4 steps describe above in **task 1**

2. In the shell type the following command to run a Node.js image in ineteractive mode. In addition expose the webservice port by using -p command line option. Once the below comand is run a blue hyperlink shall appear, which is link to the exposed port.
   `docker run -it -p 8080 node:10-slim node`

3. In the node shell type/copy paste following script and press enter
   `var http = require('http');http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/plain'}); res.end('Hello World!!!');}).listen(8080);`

   The actual code looks like the following:

```
var http = require('http');
http.createServer(function (req, res) {
        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.end('Hello World!!!');
}).listen(8080);
```

4. Click the blue hyperlink as highlighted above. It opens a new tab with output of the code above, which is "Hello World!!!".

Now lets see how this works. As evident in step 2, First we exposed a docker port using -p command line option, which docker logs conveniently exposed to us as a hyperlink. At this point we shall get an error in the browser if we click the hyperlink. The reason is that though we expose the port, there is no Application application to recieve the HTTP packets and send a response.

Hence we move on to step 3, where in we use a Javascript to listen to the port (8080) exposed in docker, and then respond on recieving a http request. Each time when a HTTP request is recieved in Port 8080, the Node server invokes the function(req, res). Consequently the code writes a plain text response "Hello World" back to the browser as HTTP response using the response object passed param res.

Task 3 Create above service using Express.js

Since there is no readymade Docker image with express pre-installed, we shall use a docker image with Node as above. Then we shall use the Node's npm command to install Express. Then we shall run the express based Javascript. The complete steps as below:

1. Follow the first 4 steps describe above in **task 1**
2. In the shell type the following command to run a Node.js image in ineteractive mode. In addition expose the webservice port by using -p command line option. Once the below comand is run a blue hyperlink shall appear, which is link to the exposed port.
   `docker run -it -p 8080 node:10-slim bash`

   Here the we use the bash shell instead of Node shell since we need to first install Express.
3. Now in the bash shell type the following
   `npm install express`
4. Now in the bash shell, open the Node interactive shell type the following
   `node`
5. In the node interactive shell copy paste the following javacript code.
   `const express = require('express'); const app = express(); const port = 8080; app.get('/',`

```
(req, res) => res.send('Hello World!!!')); app.listen(port, () => console.log(`App listening on port ${port}!`));
```

The actual code looks

```
const express = require('express');
const app = express();
const port = 8080;
app.get('/', (req, res) =>
        res.send('Hello World!!!'));
        app.listen(port, () =>
        console.log(`App listening on port ${port}!`));
```

6. Click the blue hyperlink exposed by Docker. It opens a new tab with output of the code above, which is "Hello World!!!".