Introduction Node with Express.js running on Docker

express application

In the previous section the process started by writing package.json and then running a container which would read this in and install dependencies to produce a development environment for the project.

This time the process will have an extra step, using a container to produce the package json file and a surrounding folder structure prior to running this in a separate developer container.

The process:

1. Create an express project template in a folder.
- Write a dockerfile and build an express generator image
- Run an express generator container using -v option to execute the generator and leave the generated template in a local file structure
2. Run the project template in a development environment, edit and test it.
- Write a dockerfile to build an image which will be the development environment for express
- Run a development container which will copy files in from the template project
- Edit and test the project

In this section the first two steps only are considered to set up the docker - express - nodemon platform.

1. Create an express project template in a folder

Create a new folder named express_generator and cd the terminal into this. The only thing which will be in this will be a dockerfile.

Create a dockerfile:

```
FROM node:10
ENV DEBUG="app:*"

RUN npm install -g express-generator ;\
    mkdir -p /usr/src

WORKDIR /usr/src
VOLUME ["/usr/src"]
```

docker build -t dt/express-generator .

Terminal output:

```
Sending build context to Docker daemon  2.048kB
Step 1/5 : FROM node:10
 ---> 4ae749096a47
Step 2/5 : ENV DEBUG="app:*"
 ---> Running in dc28dc7688e3
Removing intermediate container dc28dc7688e3
 ---> 294850ef8ae7
Step 3/5 : RUN npm install -g express-generator ;    mkdir -p /usr/src
 ---> Running in 05f9688f0684
/usr/local/bin/express -> /usr/local/lib/node_modules/express-
generator/bin/express-cli.js
+ express-generator@4.16.1
added 10 packages from 13 contributors in 1.073s
Removing intermediate container 05f9688f0684
 ---> 1c507f25966e
Step 4/5 : WORKDIR /usr/src
 ---> Running in a01b72e07295
Removing intermediate container a01b72e07295
 ---> 74ef3b558a16
Step 5/5 : VOLUME ["/usr/src"]
 ---> Running in 35955ff1b5fd
Removing intermediate container 35955ff1b5fd
 ---> 42db6149cad8
Successfully built 42db6149cad8
Successfully tagged dt/express-generator:latest
```

Now run the container with a command to create an express app named ‘myapp’

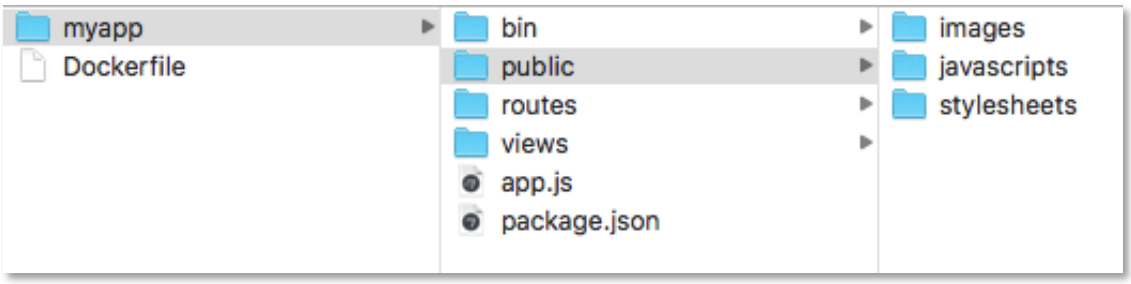
> docker run –rm -v $(pwd):/usr/src dt/express-generator express –view=pug myapp

Terminal output:

```
   create : myapp/
   create : myapp/public/
   create : myapp/public/javascripts/
   create : myapp/public/images/
   create : myapp/public/stylesheets/
   create : myapp/public/stylesheets/style.css
   create : myapp/routes/
   create : myapp/routes/index.js
   create : myapp/routes/users.js
   create : myapp/views/
   create : myapp/views/error.pug
   create : myapp/views/index.pug
   create : myapp/views/layout.pug
   create : myapp/app.js
   create : myapp/package.json
   create : myapp/bin/
   create : myapp/bin/www
```

```
change directory:
  $ cd myapp

install dependencies:
  $ npm install

run the app:
  $ DEBUG=myapp:* npm start
```

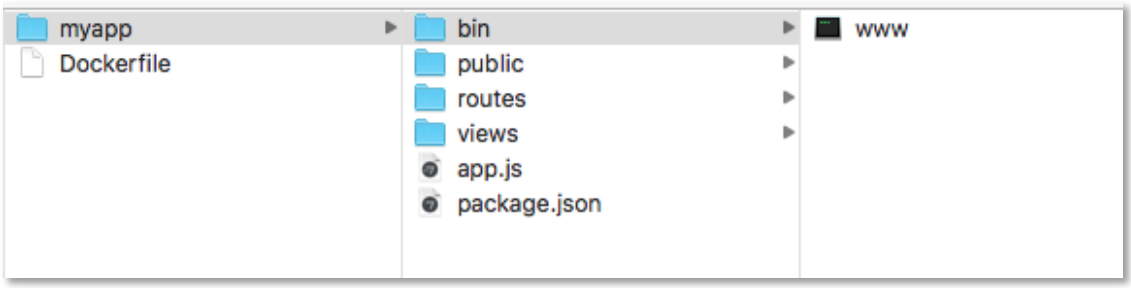
The standard files and structure are now in place in the local directory.



myapp folder

Files can be inspected.

Note the bin folder contains an executable file www



myapp folder

This is a characteristic mark of express 4. The first few lines of www are:

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('myapp:server');
var http = require('http');
```

```
/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);
```

There is more, but the point is that this is a node module which starts the app.

The package.json file shows the dependancies and starting point "node ./bin/www

```
{
  "name": "myapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "morgan": "~1.9.1",
    "pug": "2.0.0-beta11"
  }
}
```

Although it is bound to change in the future, nodemon will not start the app in this way, so some changes will need to be made to the generated files following the advice of the expressjs migration guide.

2. Run the project template in a development environment, edit and test it.

Duplicate the express-generator folder to express_app1. Change directory to this in the terminal.

Add the .dockerinore file alongside the dockerfile

```
node_modules
npm-debug.log
```

Edit package.json adding nodemon to the dependancies. Nodemon can be configured by adding a nodemonConfig section. The options added here are fairly standard, you would

need to write more if you were using a different language such as perl or processing, but javascript is a default. Take care over commas.

Nodemon cannot open ./bin/www as this is a node module not a javascript file so need the following three steps to adapt this.

Be sure to use the -L flag with nodemon, otherwise it will not respond automatically to files changed in the container.

How to use nodemon with express 4 1. Add start.js to the projects root folder 2. Edit this to

```
require('./bin/www')
```

3. Change start script entry in package.json to

```
"scripts": {
    "start": "nodemon start.js"
  },
```

package.json becomes:

```
{
  "name": "myapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "nodemon -L start.js"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "morgan": "~1.9.1",
    "pug": "2.0.0-beta11",
    "nodemon": "1.19.1"
  },
  "nodemonConfig": {
    "delay": "1500",
    "verbose": "true"
  }
}
```

Update the dockerfile to create a development environment.

Running npm install will install dependencies from package.json. The node_modules folder is moved to the root.

Nodemon is installer globally -g

```
FROM node:10
ENV DEBUG="myapp:*"
```

```
# Create app directory
RUN mkdir -p /usr/src/app
COPY  ./myapp/package.json  /usr/src/app/package.json

WORKDIR /usr/src/app
RUN npm install && mv /usr/src/app/node_modules /node_modules ;\
    npm install -g nodemon

COPY ./myapp /usr/src/app


VOLUME ["/usr/src/app"]
EXPOSE 3000
CMD [ "npm","run","start" ]
```

Build an image ready to create the development environment.

docker build -t dt/express-development .

Terminal output draws some items which I have previously pulled directly from my cache. The “changed again” message is just from my debugging and you should not see this.

```
Sending build context to Docker daemon  27.14kB
Step 1/11 : FROM node:10
 ---> 4ae749096a47
Step 2/11 : ENV DEBUG="myapp:*"
 ---> Running in 68c4effd4089
Removing intermediate container 68c4effd4089
 ---> 54d4fa7773a6
Step 3/11 : RUN mkdir -p /usr/src/app
 ---> Running in e9553168eb39
Removing intermediate container e9553168eb39
 ---> 296f1419f105
Step 4/11 : COPY  ./myapp/package.json  /usr/src/app/package.json
 ---> 3625bb6ec655
Step 5/11 : RUN echo "changed again"
 ---> Running in 3f5718b979c2
changed again
Removing intermediate container 3f5718b979c2
 ---> 80a29fc6f439
Step 6/11 : WORKDIR /usr/src/app
 ---> Running in 6ef51e4b3b34
Removing intermediate container 6ef51e4b3b34
 ---> 4548fb60c3f6
Step 7/11 : RUN npm install && mv /usr/src/app/node_modules /node_modules ;
npm install -g nodemon
 ---> Running in 7ed41f89c824

> core-js@2.6.9 postinstall /usr/src/app/node_modules/core-js
```

```
> node scripts/postinstall || echo "ignore"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for
polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open
Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a
good job -)


> nodemon@1.19.1 postinstall /usr/src/app/node_modules/nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
 > https://opencollective.com/nodemon/donate

npm notice created a lockfile as package-lock.json. You should commit this
file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9
(node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for
fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current:
{"os":"linux","arch":"x64"})

added 330 packages from 247 contributors and audited 2516 packages in 14.994s
found 1 low severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
/usr/local/bin/nodemon -> /usr/local/lib/node_modules/nodemon/bin/nodemon.js

> nodemon@1.19.1 postinstall /usr/local/lib/node_modules/nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
 > https://opencollective.com/nodemon/donate

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9
(node_modules/nodemon/node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for
fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current:
{"os":"linux","arch":"x64"})

+ nodemon@1.19.1
added 221 packages from 128 contributors in 8.969s
Removing intermediate container 7ed41f89c824
 ---> 5b63616ebe3a
```

```
Step 8/11 : COPY ./myapp /usr/src/app
 ---> d298d7d94ef2
Step 9/11 : VOLUME ["/usr/src/app"]
 ---> Running in ff7315a282db
Removing intermediate container ff7315a282db
 ---> bbb8df9d43c1
Step 10/11 : EXPOSE 3000
 ---> Running in 2138e58e13d8
Removing intermediate container 2138e58e13d8
 ---> 7f02a05039fd
Step 11/11 : CMD [ "npm","run","start" ]
 ---> Running in be1b2edd147f
Removing intermediate container be1b2edd147f
 ---> 3a27dbe5e5f5
Successfully built 3a27dbe5e5f5
Successfully tagged dt/express-development:latest
```

Now run the container named myapp1 in an interactive terminal.

The local files can be bound to the container volume using either the -v or –mount options of docker run. These lines should be equivalent:

Using -v >docker run –name myapp1 -p 3000:3000 -v $(pwd)/myapp:/usr/src/app -it dt/express-development

Using –mount (preferred)

> docker run –name myapp1 -p 3000:3000 –mount type=bind,source=$(pwd)/myapp,target=/usr/src/app -it dt/express-development

I am preferring mount because it works best for me. Volume would create a new folder in the container if one did not exist, wheras mount would throw up an error and this would be useful to prevent a quiet error from preventing problems further on.

Terminal output

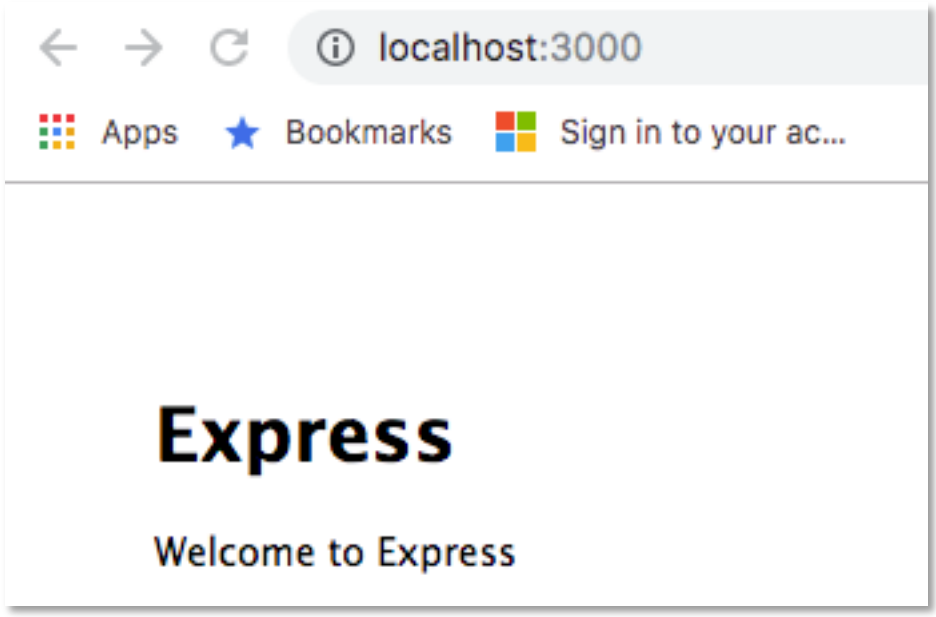
```
 > myapp@0.0.0 start /usr/src/app
> nodemon -L start.js

[nodemon] 1.19.1
[nodemon] reading config ./package.json
[nodemon] to restart at any time, enter `rs`
[nodemon] or send SIGHUP to 25 to restart
[nodemon] watching: *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node start.js`
[nodemon] forking
[nodemon] child pid: 42
[nodemon] watching 10 files
  myapp:server Listening on port 3000 +0ms
```

Direct the browser to >localhost:3000

See the request handled by express in the terminal output:

```
GET / 304 701.025 ms - -
GET /stylesheets/style.css 304 7.999 ms - -
```

Then inspect the browser output:



myapp folder

To test nodemon, change the title in routes/index.js from 'Express' to 'Express with Nodemon'

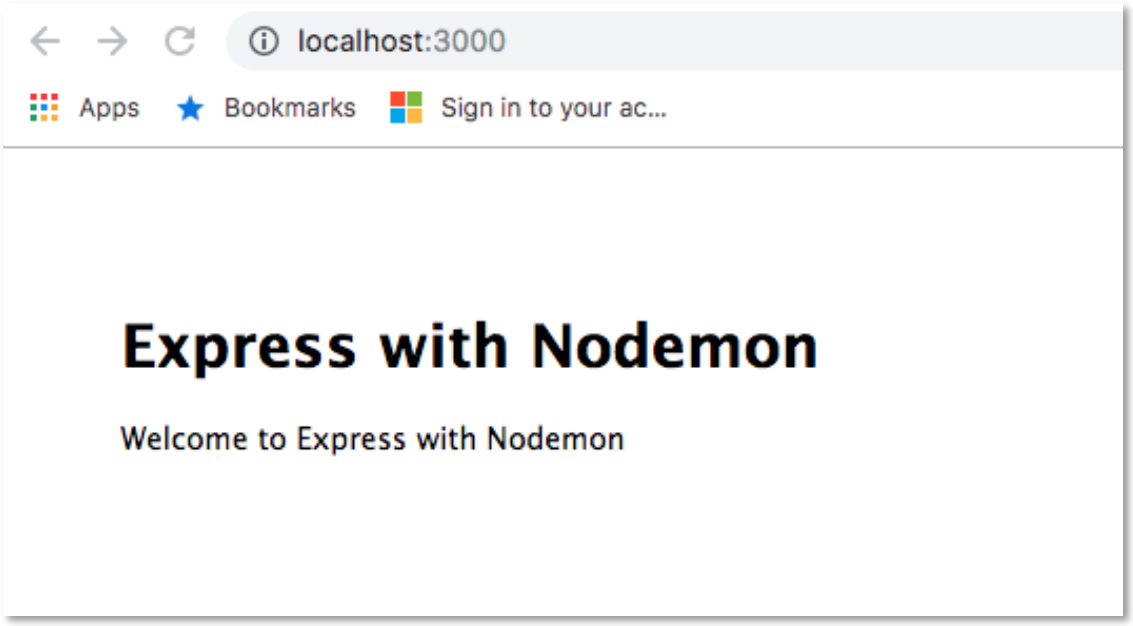
```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express with Nodemon' });
});

module.exports = router;
```

See nodemon restart the server in the terminal output when the file is saved.

```
[nodemon] files triggering change check: routes/index.js
[nodemon] matched rule: **/*.*
[nodemon] changes after filters (before/after): 1/1
[nodemon] delaying restart for 1500ms
[nodemon] restarting due to changes...
[nodemon] routes/index.js
```

```
[nodemon] starting `node start.js`
[nodemon] forking
[nodemon] child pid: 55
  myapp:server Listening on port 3000 +0ms
```

Refresh the browser and see the change in output.



changed output

As a backup, if you want to restart the server manually you can do this at any point by the restart (rs) command in the terminal.

rs

Terminal output:

```
[nodemon] starting `node start.js`
[nodemon] forking
[nodemon] child pid: 66
  myapp:server Listening on port 3000 +0ms
```

now refresh the browser.

Close the server by

CTRL + C

Then stop and remove the container and generally tidy up.

references

How To Build a Node.js Application with Docker sharks webpage

express

How to create a REST API with Express.js in Node.js