

## Project One

### Objectives

The first objective of this project was to create a local application that would let you enter your name and location and save this information to a database. The local app would be in a VirtualBox virtual machine, using a CentOS operating system. The app would use Express.js as a framework and pug to create the frontend user interface. The application was local so should be accessible in a browser via <http://localhost>, this would require a reverse proxy configured with Nginx. The database would use MongoDB to store and present the information entered.

Once the app was made, the next objective was to use Docker to containerize the application, this would require a Dockerfile to be written for the app and the database. Then using a Docker-compose file to make all the containers run simultaneously. These containers would need to communicate with each other so that the information was sent to the database.

Lastly the app would need a proxy pass for port 443, this is the https:// port. This required SSL certificates and would make the app accessible via <https://localhost>.

### Solution

I started by creating a CentOS virtual machine. This was done by downloading a package and launching the VM using that. Then I installed the necessary prerequisite programs onto the machine. Once this was done I began by writing an app.js and start.js file, which defined what files and programs the app would run (including express and mongodb). This would include running a index.pug file which I wrote the user interface in. Once this was done I used the “npm run watch” input to run my app and make it listen on port 3000, then I could port forward the host ip port 80 to the VM port 3000, this would then mean that <http://localhost> would work. I also port forwarded host 27017 to VM port 27017, as this is the port for the mongodb database.

Now that I had my app running and working I wrote a Dockerfile for the app, in this file I defined a platform (node was used for this app), running an update and upgrade function to ensure the software is up to date, and a function to install npm and pug. Then exposes the container to port 3000 and runs “npm run watch” to run the app. I also wrote a dockerfile for the proxy pass, this used a nginx base image and ran a reverse-proxy.conf file which I wrote to listen to port 443 and pass it through my localhost:3000.

Next I would need to write a docker-compose file which would run all these docker containers simultaneously, in this file I defined my services, nginx, app, and mongodb, the image each of these would use, the ports each would communicate on and the volumes they would need to operate. Once I had successfully done this I could type “docker-compose run”

and it would run this file and my app would be working, listening on port 443 and have a database.

### Problems and solutions

My first problem was trying to download the programs on my virtual machine, I would get a connection error when I first tried, this was because my virtual machine was not communicating with the internet. To fix this I disabled the firewall (fine on a local machine) and also added some port forwarding routes in my virtualbox settings.

My next problem occurred when I started to create the user interface in my pug file. I wrote the code in usual HTML, but pug uses its own syntax, luckily it is essentially a simplified version of HTML and with a little research I was able to translate my HTML code to pug format.

I had multiple syntax errors along my way, I found the best solution to this was to read the error messages on my terminal and try to pin-point where I had gone wrong. Then re-read my code to see if I had missed something. Eventually they all got resolved.

My main problem occurred when writing the reverse-proxy.conf file, this is due to needing ssl certificates and the related ssh keys. I found a document online which outlined how to create the ssl certificates and one of my teammates helped me by telling me that I could mount that into my nginx docker container. This now meant my container had the needed SSL certificates to communicate on port 443 and https:// would work. Results

```
> nodemon ./start.js
[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./start.js`
Express is running on port 3000
```


This is my app running locally just using “npm start watch” this is the result I wanted before I started on my dockerfiles. This would give me the first app on browser which looked like this:



← → ↻ ⓘ localhost:3000

FirstName:  Surname:  Location:

As you can see this is on port 300 and normal http. From this I can write my dockerfiles and proxy-pass to get the following result when “docker-compose run” is operated:



← → ↻ ⚠ Not secure | localhost

FirstName:  Surname:  Location:

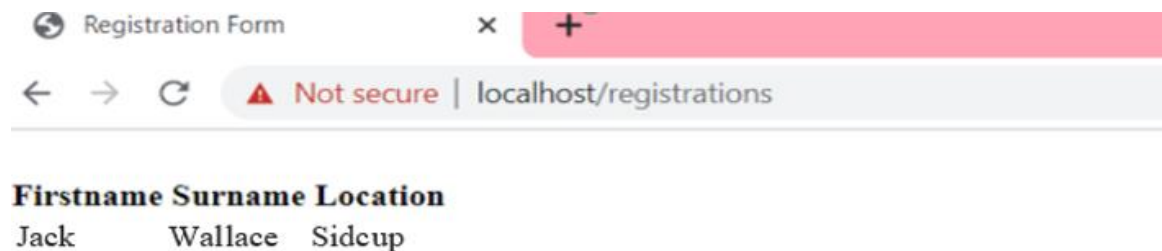
Here is the application in browser, this is the finished result, the not secure warning shows that it is on port 443 and there is now no port 3000 thanks to port forwarding.

I can then enter information like so:



FirstName:  Surname:  Location:

And then go to my database at either localhost:27017 or localhost/registrations I can see it saved to my database:



Registration Form x +

← → ↻ ⚠ Not secure | localhost/registrations

Firstname	Surname	Location
Jack	Wallace	Sidcup