SOFE 4610U:

Internet of Things

Lab #5: Using Docker for Automated System Deployment on the Raspberry Pi

Objectives

- Understand the design goals of Docker
- Setup Docker on the Raspberry Pi
- Download and run a Docker image from Docker Hub
- Design your own custom Docker Image for automated deployment of your IoT solution

Important Notes

- Work in groups of **two or three** students
- All reports must be submitted as a PDF on blackboard, if source code is included submit your report as PDF and source files as an archive (e.g. zip, tar.gz)
- Save the submission as <lab_number>_<first student's id> (e.g. lab1_100123456.pdf)
- If you cannot submit the document on blackboard then please contact the TA with your submission: **michael.lescisin@uoit.net**

What is Docker?

The slogan for Docker is “Build, Ship, and Run Any App, Anywhere”. This is the goal of Docker – to package an application, and all its dependencies so that it can be deployed to a wide variety of system architectures, easily. For example, a web application may use Apache HTTP server, MySQL database, and PHP scripting language. Traditional setup of this web application is complicated as it requires the installation of all individual components and their manual configuration. For example, when installing a web application, one may have to install the MySQL database server and then run a script to setup the required tables in that database. Docker simplifies this process; by building the application and all its dependencies into an independent Linux container, it can easily be moved from platform to platform without worrying about conflicts with the software installed on the host.

Lab Activity

Software Setup

1. Setting up Docker on the Raspberry Pi is very easy now that Docker officially supports the Raspberry Pi. To do so run the following commands:

```
curl –sSL https://get.docker.com –o get-docker.sh
chmod u+x get-docker.sh
./get-docker.sh
```

2. Once the install script finishes executing, we must now add the user *pi* to the *docker* group so that we, as non-root, may access the Docker engine. To do so run the command:

```
sudo usermod –aG docker pi
```

   Then logout and log back in as *pi*.

3. Once Docker is installed, we will run the *rpi-raspbian* Docker image from Resin.io but first the image needs to be downloaded from the Docker Hub. To do so run the command:

```
docker pull resin/rpi-raspbian
```

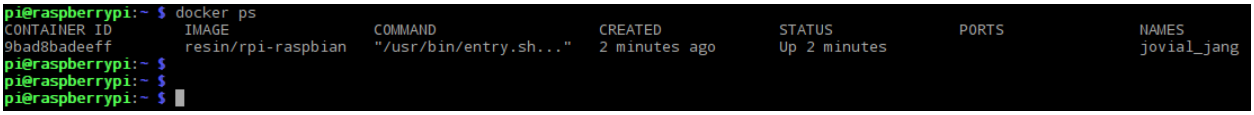
4. We will now start the image and obtain an interactive bash shell. To do so run the command:

```
docker run -it resin/rpi-raspbian
```

5. We will now list the running Docker containers. To do so open a new terminal to the Raspberry Pi and execute the command:

```
docker ps
```

In our example, the Docker engine has named our container “jovial_jang”. The name assigned to your container will almost certainly be different. Make note of the assigned container name.



6. To copy a file into the Docker container execute the command:

```
docker cp myFile jovial_jang:/
```

7. To copy a file out of the Docker container execute the command:

```
docker cp jovial_jang:/myFile .
```

Demonstrate to your lab instructor that you are able to download the Docker Raspbian image, start a container, and copy files into and out of the container.

8. We will now write a **Dockerfile** that will setup a server which will read data from the CheerLights channel on ThingSpeak and set the background of a webpage accordingly. The files implementing the server are available on Blackboard [flask_server.py , webpage_template.html].

9. Create, and switch into, a new directory named “LightServer” on the Raspberry Pi by executing the commands:

```
mkdir LightServer
cd LightServer
```

10. Copy the file **flask_server.py** into this directory. Create a sub-directory named **templates** and copy **webpage_templage.html** into that directory.

11. Create a new file named **Dockerfile** and edit it with the commands:

```
touch Dockerfile
nano Dockerfile
```

12. Add the following lines to the **Dockerfile**:

```
FROM resin/rpi-raspbian
RUN apt-get update
RUN apt-get -y install python python-requests python-flask
RUN mkdir /app
RUN mkdir /app/templates
COPY flask_server.py /app
COPY templates /app/templates
WORKDIR /app
ENTRYPOINT ["python"]
CMD ["flask_server.py"]
```

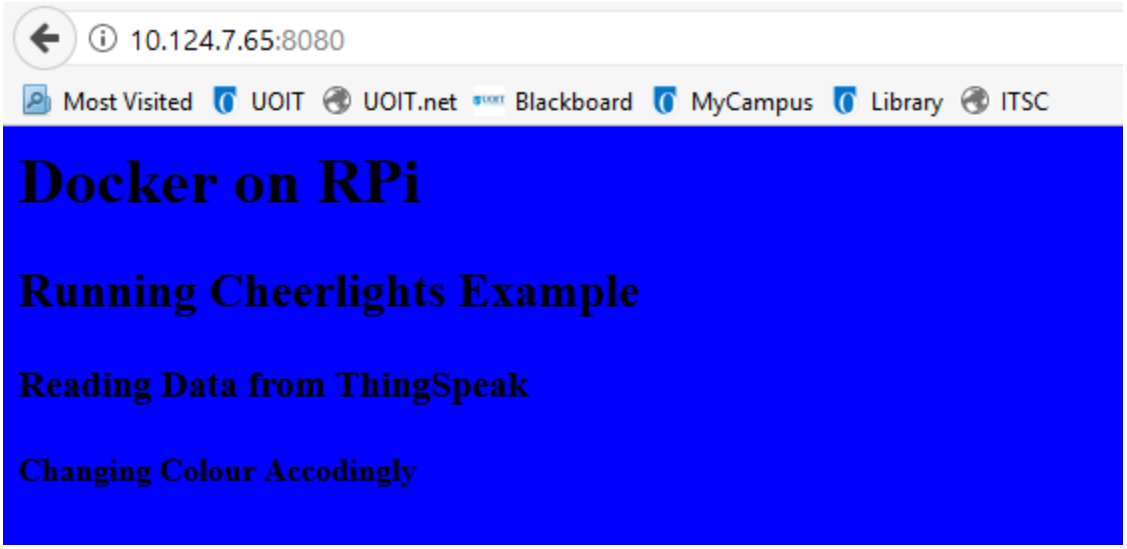
13. We will now build our new Docker image with the command:

```
docker build –t flask-cheerlights .
```

14. We will now run the image with the command:

```
docker run -p 8080:5000 –it flask-cheerlights
```

15. Load the webpage in the browser by visiting http://IP_OF_PI:8080/ replacing **IP_OF_PI** with the IP address of the Raspberry Pi which you used for SSH access.



Lab Task

Based on what you have learned from the Software Setup section of this lab, as well as the content of the previous labs, build your own IoT project that satisfies the following requirements.

- Docker is used for automated deployment.
- Networking is used. (Possible examples include MQTT, HTTP)
- There is input/output interaction with the Sensorian shield (LED, Light Sensor, etc…).
    - TIP: Run Docker with the flags `--cap-add SYS_RAWIO --device /dev/mem`
    - TIP: GPIO access in Python requires installing the packages: ***python python-rpi.gpio***
    - TIP: The **SensorsInterface** library can be installed into the Docker container by following the procedure in lab 4, but first:
        - Install the package ***build-essential***
        - Do not use ***sudo***, for example **sudo apt-get update** becomes **apt-get update**

- Please get your project idea approved with the lab TA before proceeding to building it.

Deliverables

Complete the lab task and document with step-by-step instructions such that a person with only basic knowledge of the Raspberry Pi could reproduce your work. Use of screenshots is highly recommended. Explain the problem which your IoT solution attempts to solve, as well as how the technology you used has allowed you to implement this solution (eg. the Apache Web Server has allowed the user to read the temperature from their web browser.).

Please note, all lab reports will have title pages, introduction, content of the lab tasks and conclusion.