Project 1: Docker and WebServices

Due: THURSDAY 2018 March 29 @ 11:55pm

You will have created a WebService in HW02 and Weather APP in HW03. This Project will have you port (migrate) that WebService into a docker container instance, that acts as a more portable and modular framework for hosting services.

Contrasted with the HW assignments, this project **does not** require you to remotely serve your Docker application. However, you will need to provide access to the Docker image to the professor.

Docker provides a stand-alone Linux environment where you can install your libraries and run your applications within an isolated environment. As you've installed your application in AWS already, you should be able to install it into Docker.

There are five components to this assignment:

1. On your own Linux system, within a VirtualBox VM, on an AWS, or other VM or system, install Docker. You are even fine to use the AWS docker environment, if you wish to try.
    a. `apt-get install docker.io` or `yum install docker-io`
    b. Install (as root): yum install docker-io
    c. Start the docker service (as root): service docker start
    d. Let regular users access docker (as root):
    `usermod -a -G docker <username>`
    On some machines I've had to add write permission to the socket:
    `chmod a+w /var/run/docker.sock`
2. Play with docker, experiment:
    a. https://docker.github.io/engine/getstarted/
    b. https://prakhar.me/docker-curriculum/
    c. `docker search centos`
    d. `docker pull centos`
    e. `docker images`
    f. `docker run -it <image name> <binary to run>`
    g. `docker run -d -p 8080:80 <image name> <binary to run>`
    h. `docker ps -a`
    i. `docker save <image id> > out.tar`
    j. `docker import out.tar`
    k. `docker save <image id> | gzip > out.tar.gz`
3. Install your WebService application from HW02 & HW03 into your docker project, either by deploying it directly or utilizing **git** and the **Dockerfile** to get it installed
4. Commit your Docker project into GitHub, making sure to include the README.md from your WebService in the root of the docker project. I recommend creating a new github project for this.
5. Package up your Docker container such that when it starts, it runs /startme.sh and the webservice you wrote will be listening from port 80 inside the Docker container
6. Save your docker container image (a .tar, .tgz, .zip, or .tar.gz file, normally) using "docker save".

This will create an archive that you should either upload to blackboard or to some file hosting service like Google Drive for your Blackboard submission. Please use compression.

You will be graded on the following attributes:

- README.md accurately describes the service
- Docker instance works
    - I will run "`docker load -i yourdocker.tgz`" and
    - "`docker run -d -p 8081:80 yourdocker /startme.sh`" to get it running, or
    - "`docker run -d -p 8081:80 yourdocker`"
- WebService works inside Docker identically to AWS

Submit to Canopy the following things:

- GitHub project for your Docker project with README.md
- Provide the URL to the link to download it from a file hosting site (like Google Drive, etc.) or docker.io user/image for doing docker pull
- Command to run your docker (i.e if it needs startle.sh or not)

Notes/Hints:

For deploying **Java/Tomcat WAR** file applications, the following URL may be helpful:

http://codeomitted.com/deploy-war-file-to-docker-image/

startme.sh

The following is a bash script containing a number of commented-out recipes for various Docker application-server configurations. Apache/PHP/MySQL, Python Django, Services configured to auto-start in background, Tomcat in foreground are some approaches.

```
#!/bin/bash

# mod_php, mod_python, apache, with MySQL running in background
# /usr/sbin/mysqld &
# httpd -DFOREGROUND -k start

# If services are used to start up application, without running a startup
# script
#while /bin/true; do
#  sleep 10
#done

# Tomcat, foreground commandline mode, if desired
#/usr/local/tomcat/bin/catalina.sh run

# Python django
#cd myapp
#python manage.py runserver 0.0.0.0:80
```

More below...

Dockerfile

A sample Dockerfile. In this case, you start with a base image of “mysql”, and then you build on that by installing `python-pip` and `apache2`. Then copy your startme.sh into the container and set the “run on creation” instance CMD.

```
FROM mysql

RUN apt-get update
RUN apt-get install -y python-pip
RUN apt-get install -y apache2
#RUN pip install
#RUN mysqld &
#RUN mysql < init.sql

COPY startme.sh /

CMD ["/bin/bash", "/startme.sh"]
```

Please see class wiki for creating a docker app with flask/ngnix configuration.