**CMPE 282**

**HW2**

**Student ID:** 010034700

**Student Name:** Keertikeya Gupta

**Soln 1.**

Dev Tools: Eclipse EE Luna 4.4.1 on Ubuntu 14.04

REST Client: HTML Page

REST Server: web/app component: servlet, Tomcat 7.0

REST Server (NoSQL) database component: MongoDB on Ubuntu

**Soln 2.**

Host 1: Ubuntu, 192.168.232.145, Ubuntu 14.04

Host 2: Ubuntu, 172.17.0.1, Ubuntu 14.04

**Soln 3.**

To deploy the Database Container (dbKeertikeyaGupta700):

To create the container for the MongoDB database, first we ran the Docker image for Mongo using the following command:

**docker run –it mongo**

This will download the Mongo Docker image as it is the first time we are running the image. This command creates a Mongo container with ID “b46db2d987eb”. The purpose to create this image was to allow some linking to the primary Mongo Container that we’ll be using in this asignement.

Next, we created our primary docker container as follows:

**docker run –it --name=dbKeertiekyaGupta700 --link=b46db2d987eb:mongo mongo /bin/bash**

In this container, we can run the mongo shell with the following command:

**mongo $MONGO\_PORT\_TCP\_27017\_ADDR**

For the Web container, we created our own Docker image from the Tomcat image. For this we made and used a Dockerfile whose contents are as follows:

**FROM tomcat:8.0**

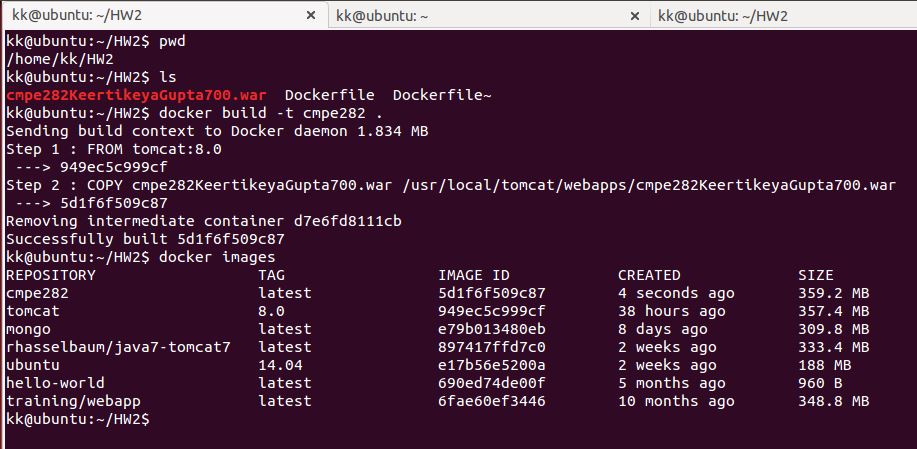
**COPY cmpe282KeertikeyaGupta700.war /usr/local/tomcat/webapps/cmpe282KeertikeyaGupta700.war**

Here, cmpe282KeertikeyaGupta700.war is the .war file exported from the Eclipse project. This file contains the necessary web application archives that will be used by the container that we will be creating.

Now for the Docker image, we use build command:

**docker build -t cmpe282 .**

For this command to run, we must be in the directory where the Dockerfile is present. The “.” At the end represents that we want to create the image in the present directory.



Once this image has been created, we can use it to create containers. We use the following command to create our web container:

**docker run –itd --name=appKeertikeyaGupta700 --link= dbKeertiekyaGupta700:db cmpe282**

This will create a new container called “appKeertiekyaGupta700” using th ecmpe282 image.

**\*\*Note:** we do not need to explicitly expose any ports because we are using the Tomcat image which already has port number 8080 exposed.

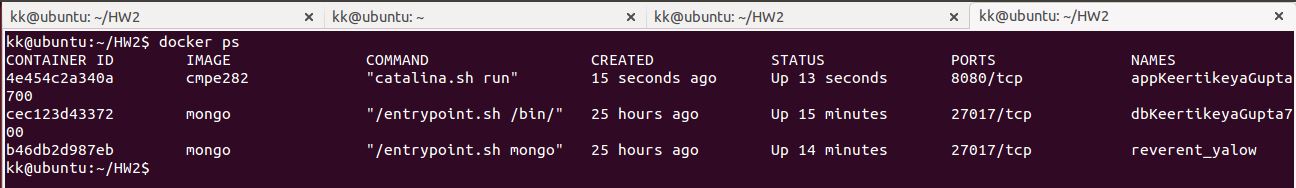
In the Servlet of our code, we need to update the MongoClient connect. Instead of using localhost, we have to specify the IP address of the DB container (172.17.0.3).

Also, we had to modify the HTML code for the Index.html file. We have changed the form submit action from “http://localhost:8080/DemoServlets/DemoServlet” to

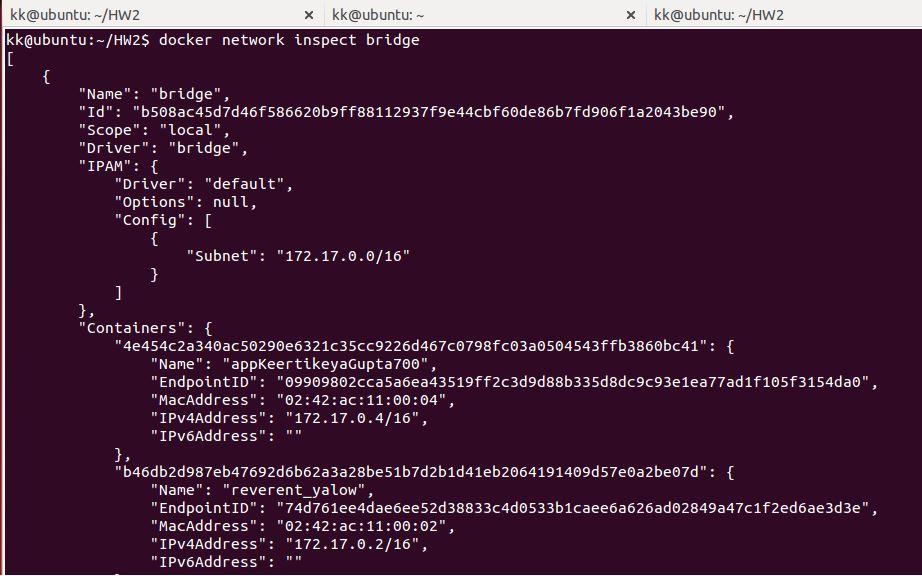
<http://172.17.0.4:8080/cmpe282KeertikeyaGupta700/WEB-INF/classes/com/demo/kk/DemoServlet>

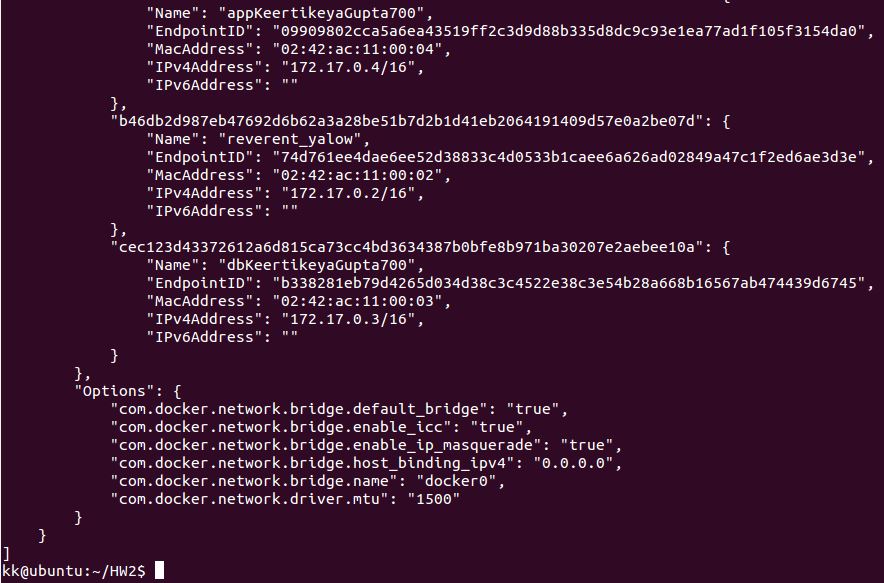
**Soln 4.**

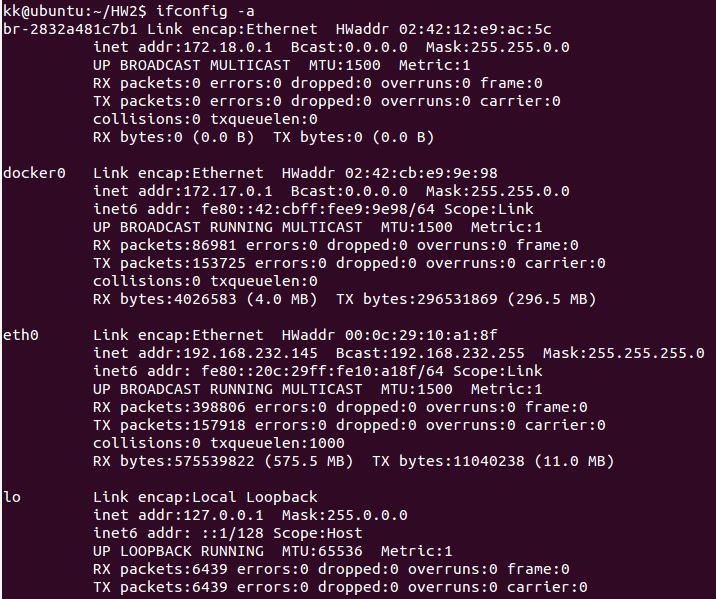
docker ps command shows the list of running containers

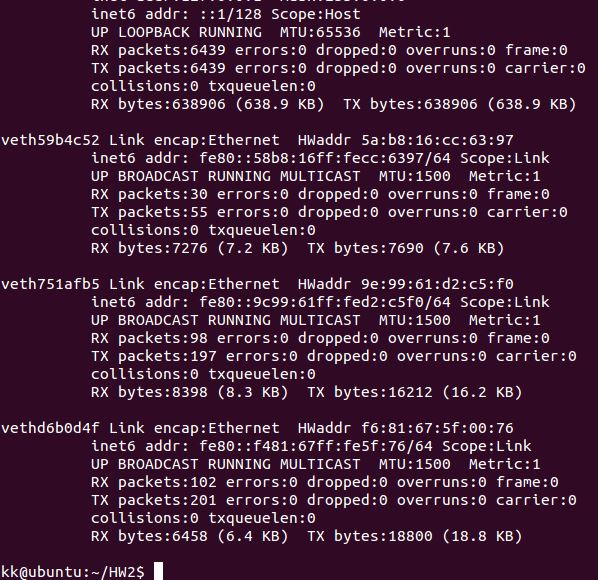


By default, when we create a new container, it always goes to the bridge network. The **docker network inspect bridge** shows the details of the bridge network:





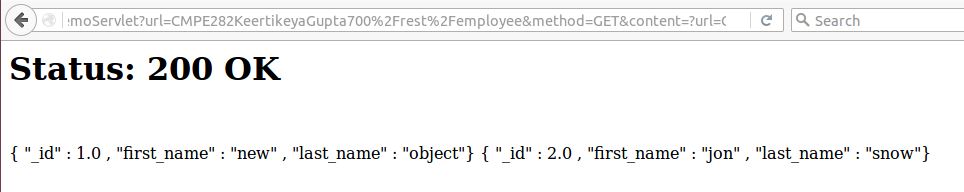




**Soln 5.**

Screenshots:

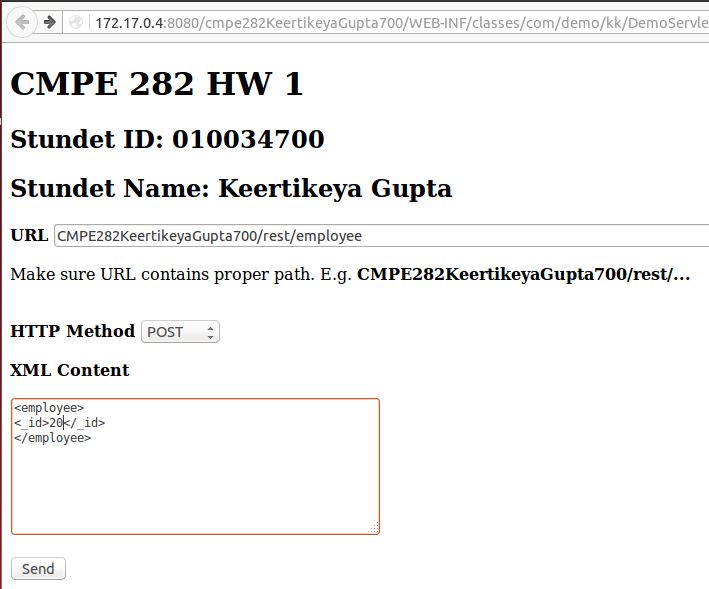
First, we had already added two documents in the database:

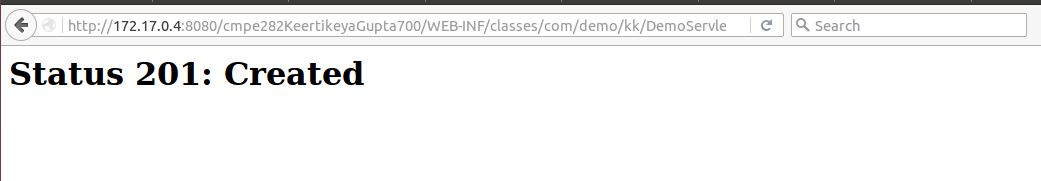


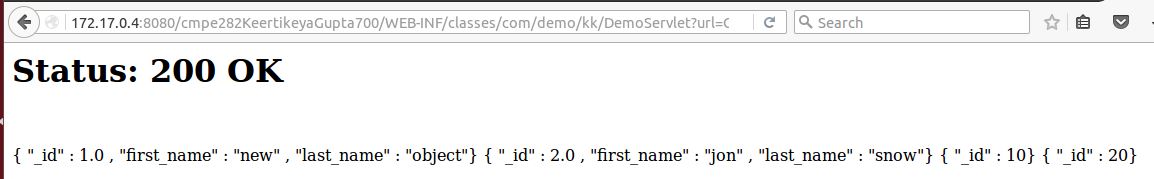
To create employee with ID = 10 with POST:



Creating employee with ID 20 using POST:







Using the PUT method to update the data of employee with ID=10

PUT method to update employee ID = 20



Retrieving the updated data with GET:  


Deleting the document for employee ID = 20:

GET method to retrieve all data: