HOMEWORK 2

Akshay Mishra **011476673**

Q1.

a. List technologies, software (including version), and platform (include version) for the REST client, REST server, and NoSQL.

Answer:

Docker:

```
Client:
 Version:      17.09.0-ce
 API version:  1.32
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:41:23 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.09.0-ce
 API version:  1.32 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:42:49 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

on

Linux centos7desktop 3.10.0-123.el7.x86_64 #1 SMP Mon Jun 30 12:09:22 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

Docker-compose version (Extra Credit):

```
docker-compose version 1.16.1, build 6d1ac21
```

docker-py version: 2.5.1
CPython version: 2.7.13
OpenSSL version: OpenSSL 1.0.1t 3 May 2016
on

Linux centos7desktop 3.10.0-123.el7.x86_64 #1 SMP Mon Jun 30 12:09:22 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

REST Client:

curl 7.54.0 (x86_64-apple-darwin16.0) libcurl/7.54.0 SecureTransport zlib/1.2.8
on
MacOS Sierra 10.12.6

And

curl 7.29.0 (x86_64-redhat-linux-gnu) libcurl/7.29.0 NSS/3.15.4 zlib/1.2.7 libidn/1.28 libssh2/1.4.3
on
Centos 7 Desktop v3.10

REST Server:

Embedded Tomcat 8.5 in Spring Boot Jar inside a docker image
on
Centos 7 Desktop v3.10 (Host 1) &
MacOS Sierra 10.12.6 (Host 2)

b. For host1 and host2, list their OS (and version), and IP address.

Answer:

Host 1:

OS: Centos
Linux centos7desktop 3.10.0-123.el7.x86_64 #1 SMP Mon Jun 30 12:09:22 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

IP Address:

(IP Address as per dashboard SJSU VSphere)

130.65.159.120

Host 2:

OS: OSX
MacOS Sierra 10.12.6

IP Address:

When connected from Home Wi-fi to SJSU network via VPN

10.0.0.68

c. A sample entire HTTP URL (including actual IP address of host1), URI, and request body for POST to create a new `employee` based on the XML format or JSON format (depending on your implementation). Also indicate if there is any additional setup (e.g., HTTP header, etc.).

Answer:

entire URL:

http://130.65.159.120:8080/cmpe282Akshay673/rest/employee

URI: `POST /cmpe282Akshay673/rest/employee`

Request body:

```
{
        "id": 1,
        "fname": "Akshay",
        "lname": "Mishra"
}
```

no additional setup (no additional request header) is needed.

Q2. Describe detailed steps to **build** webapp and db docker containers with screenshots. (You can pull docker images directly, utilize Dockerfile to build one, or build on your own, or combination of the above.)

Answer:

WebApp - My Own Docker Image:

For the web app, I built my own image and deployed JAR file inside the container. The Jar runs an embedded tomcat hence one doesn't need to install Tomcat or any Spring dependencies, any of these in order to run the webapp.

Base Image: openjdk:8-jdk

Modifications:

Added my cmpe282Akshay673.jar file to the home directory in order to use it to run the app.

et ENTRYPOINT as the command that calls the JAR file and passes appropriate parameters that override the original ones in the JAR in order to mainly locate MongoDB Host, Port and URL. I leveraged the networking features of Docker here while passing the parameter.

Dockerfile for WebApp:

```
FROM openjdk:8-jdk

MAINTAINER akshay.mishra@sjsu.edu

ADD cmpe282Akshay673.jar  cmpe282Akshay673.jar

RUN sh -c 'touch /cmpe282Akshay673.jar'

ENTRYPOINT ["java", "-Dmongodb.host=dbAkshay673","-
Dmongodb.db=cmpe282Akshay673","-Dmongodb.port=27017","-
Djava.security.egd=file:/dev/./urandom","-
jar","/cmpe282Akshay673.jar"]
```

Explanation:

```
FROM openjdk:8-jdk
```

This basically is just saying that we are inheriting the openjdk v8 as our base image on top of which we are building our own image.

```
MAINTAINER akshay.mishra@sjsu.edu
```

This is an optional tag and used to identify the owner of the docker image. People may use this in order to contact the owner in case they have any suggestions or issues.

```
ADD cmpe282Akshay673.jar  cmpe282Akshay673.jar
```

This file basically adds the JAR to the container in its home directory. This used JAR will be called via java –jar command in order to run the web app.

```
RUN sh -c 'touch /cmpe282Akshay673.jar'
```

This command just runs ‘touch’ on the jar which only modifies the last accessed date of the JAR.

```
EXPOSE 8080
```

This command is basically exposing the port 8080. Then we can use it to map later with a port on the host. By default, 8080 is open and hence we won’t have any issues even if we omit this sentence. For a port that is not open by default, it must be exposed explicitly in the dockerfile in order to be able to map it with a host port.

```
ENTRYPOINT ["java", "-Dmongodb.host=dbAkshay673","-
Dmongodb.db=cmpe282Akshay673","-Dmongodb.port=27017","-
Djava.security.egd=file:/dev/./urandom","-jar","/cmpe282Akshay673.jar"]
```

This line is the most important line in the dockerfile. This basically sets the entrypoint of the container. It means that the container will run this command when it is up. This command in essence runs the JAR file using java –jar command and spins up the app.

Docker image for the DB

Image: official MongoDB Docker Image (v3.15) from hub.docker.com

Modifications: No modifications made to the original image, only performed volume mapping on the container.

Steps to build WebApp and MongoDB Containers:

1. **Spin up MongoDB Container using below command:**

```
docker run -d -v data:/data/db --name dbAkshay673  mongo
```

This will pull/use existing (if previously pulled) official image of mongodb and create a container with the name as dbAkshay673

Here, we are also mapping /data/db of the container to data directory on the host. This is called as volume mapping.

2. **Build WebApp Image using below command:**

```
docker build --no-cache=true -t app_akshay_673 .
```

Run this from the directory where Dockerfile for the WebApp Image is stored. This should build an image with the name as app_akshay_673.
Once image is built, then the webapp container can be created using below command:

```
docker run -it --name appAkshay673 --link dbAkshay673:dbAkshay673 -p 8080:8080 app_akshay_673
```

This should create a container named appAkshay673 and automatically map port 8080 of the container to the port 8080 of the host.

Known Issues: None

Additional features You are proud of:

1. I am using link feature of Docker, which helps to naturally induce environment variables in the target container from the source container. These env variables are exposed from the source container. These can be used to make these two containers communicate with each other using the default bridge network.

2. I am using volume mapping on the MongoDB container to persist the data on the local host (outside the container). Even in case of calamity, if the MongoDB container stops, it can be restarted from the state where it had stopped using this persisted data on the host. Hence the system is more resilient now.

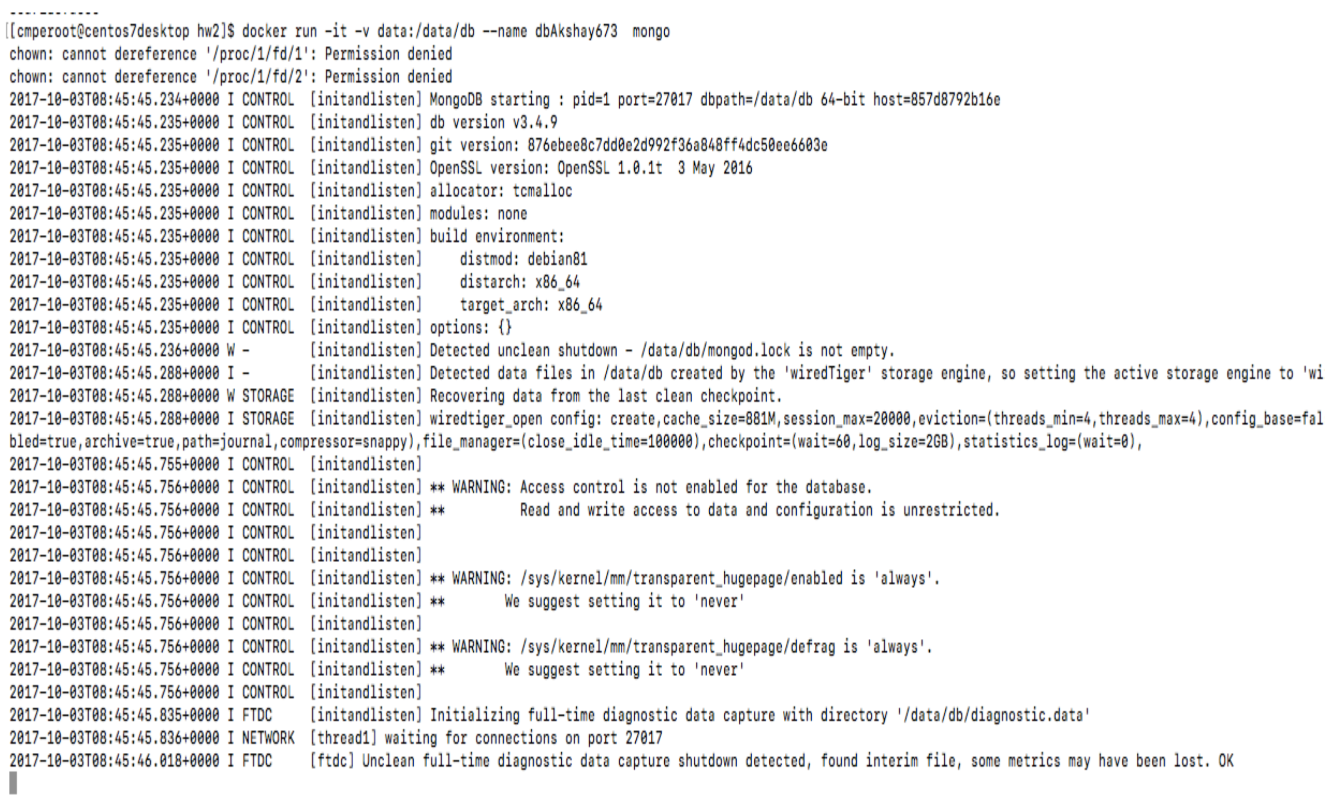
3. I am overriding the application properties in the JAR command and simply passing some environment variables that were induced with the help of ‘link’ tag. No configuration is hardcoded and hence system is more easy to configure and use.

Q3. Describe detailed steps to **deploy** webapp and db docker containers with screenshots.

Answer:

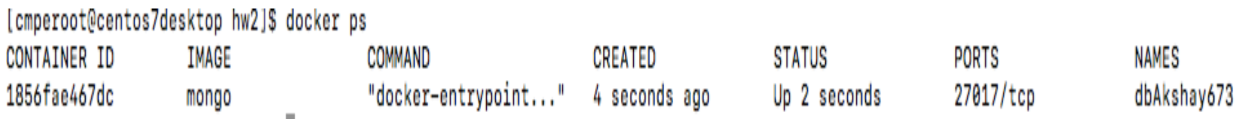
Docker command to deploy MongoDB container:

docker run -d -v data:/data/db --name dbAkshay673 mongo



```
[[cmperoot@centos7desktop hw2]$ docker run -it -v data:/data/db --name dbAkshay673  mongo
chown: cannot dereference '/proc/1/fd/1': Permission denied
chown: cannot dereference '/proc/1/fd/2': Permission denied
2017-10-03T08:45:45.234+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-bit host=857d8792b16e
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] db version v3.4.9
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] git version: 876ebee8c7dd0e2d992f36a848ff4dc50ee6603e
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1t  3 May 2016
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] modules: none
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] build environment:
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten]     distmod: debian81
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten]     distarch: x86_64
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten]     target_arch: x86_64
2017-10-03T08:45:45.235+0000 I CONTROL  [initandlisten] options: {}
2017-10-03T08:45:45.236+0000 W -        [initandlisten] Detected unclean shutdown - /data/db/mongod.lock is not empty.
2017-10-03T08:45:45.288+0000 I -        [initandlisten] Detected data files in /data/db created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wi
2017-10-03T08:45:45.288+0000 W STORAGE  [initandlisten] Recovering data from the last clean checkpoint.
2017-10-03T08:45:45.288+0000 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=881M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=fal
bled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-10-03T08:45:45.755+0000 I CONTROL  [initandlisten]
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten]
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten]
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten]
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten] **        We suggest setting it to 'never'
2017-10-03T08:45:45.756+0000 I CONTROL  [initandlisten]
2017-10-03T08:45:45.835+0000 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory '/data/db/diagnostic.data'
2017-10-03T08:45:45.836+0000 I NETWORK  [thread1] waiting for connections on port 27017
2017-10-03T08:45:46.018+0000 I FTDC     [ftdc] Unclean full-time diagnostic data capture shutdown detected, found interim file, some metrics may have been lost. OK
```

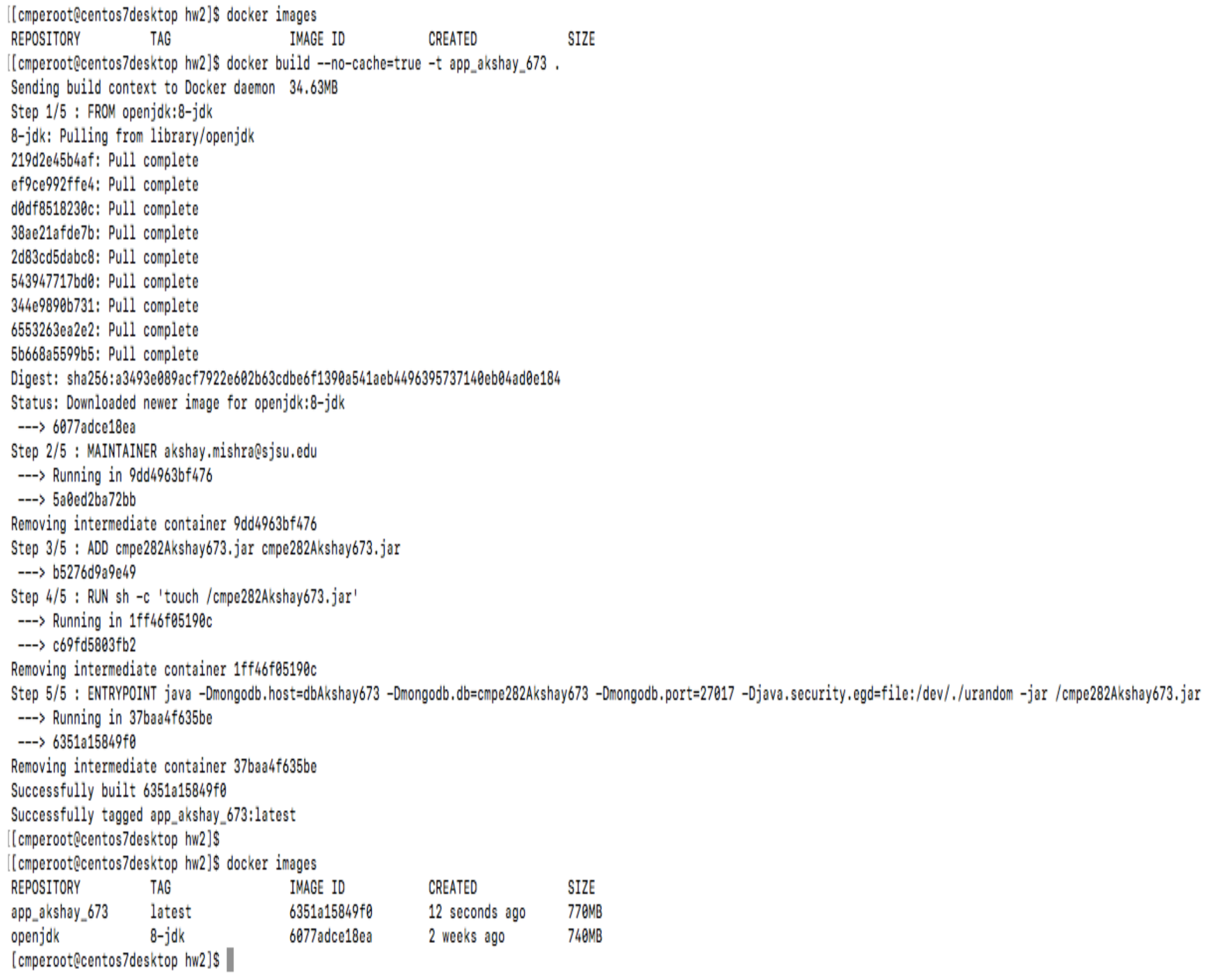
docker ps



```
[cmperoot@centos7desktop hw2]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
1856fae467dc        mongo               "docker-entrypoint..."   4 seconds ago       Up 2 seconds        27017/tcp           dbAkshay673
```

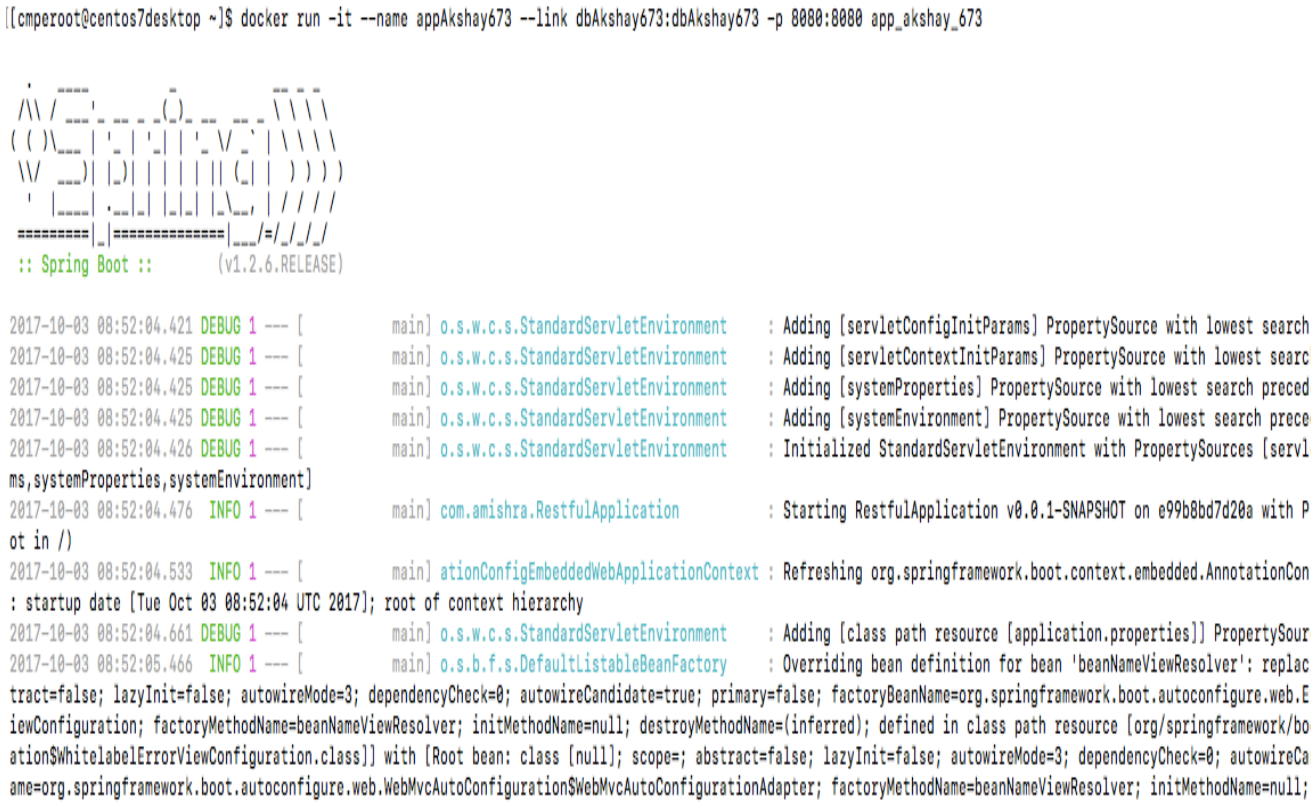
Docker command to build Web App:

docker build --no-cache=true -t app_akshay_673 .



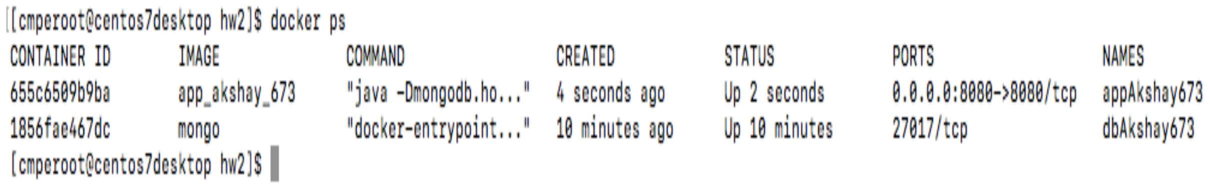
```
[[cmperoot@centos7desktop hw2]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[[cmperoot@centos7desktop hw2]$ docker build --no-cache=true -t app_akshay_673 .
Sending build context to Docker daemon  34.63MB
Step 1/5 : FROM openjdk:8-jdk
8-jdk: Pulling from library/openjdk
219d2e45b4af: Pull complete
ef9ce992ffe4: Pull complete
d0df8518230c: Pull complete
38ae21afde7b: Pull complete
2d83cd5dabc8: Pull complete
543947717bd0: Pull complete
344e9890b731: Pull complete
6553263ea2e2: Pull complete
5b668a5599b5: Pull complete
Digest: sha256:a3493e089acf7922e602b63cdbe6f1390a541aeb4496395737140eb04ad0e184
Status: Downloaded newer image for openjdk:8-jdk
 ---> 6077adce18ea
Step 2/5 : MAINTAINER akshay.mishra@sjsu.edu
 ---> Running in 9dd4963bf476
 ---> 5a0ed2ba72bb
Removing intermediate container 9dd4963bf476
Step 3/5 : ADD cmpe282Akshay673.jar cmpe282Akshay673.jar
 ---> b5276d9a9e49
Step 4/5 : RUN sh -c 'touch /cmpe282Akshay673.jar'
 ---> Running in 1ff46f05190c
 ---> c69fd5803fb2
Removing intermediate container 1ff46f05190c
Step 5/5 : ENTRYPOINT java -Dmongodb.host=dbAkshay673 -Dmongodb.db=cmpe282Akshay673 -Dmongodb.port=27017 -Djava.security.egd=file:/dev/./urandom -jar /cmpe282Akshay673.jar
 ---> Running in 37baa4f635be
 ---> 6351a15849f0
Removing intermediate container 37baa4f635be
Successfully built 6351a15849f0
Successfully tagged app_akshay_673:latest
[[cmperoot@centos7desktop hw2]$
[[cmperoot@centos7desktop hw2]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
app_akshay_673      latest              6351a15849f0        12 seconds ago      770MB
openjdk             8-jdk               6077adce18ea        2 weeks ago         740MB
[cmperoot@centos7desktop hw2]$
```

docker run -it --name appAkshay673 --link dbAkshay673:dbAkshay673 -p 8080:8080 app_akshay_673



```
[[cmperoot@centos7desktop ~]$ docker run -it --name appAkshay673 --link dbAkshay673:dbAkshay673 -p 8080:8080 app_akshay_673

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.2.6.RELEASE)

2017-10-03 08:52:04.421 DEBUG 1 --- [           main] o.s.w.c.s.StandardServletEnvironment     : Adding [servletConfigInitParams] PropertySource with lowest search
2017-10-03 08:52:04.425 DEBUG 1 --- [           main] o.s.w.c.s.StandardServletEnvironment     : Adding [servletContextInitParams] PropertySource with lowest searc
2017-10-03 08:52:04.425 DEBUG 1 --- [           main] o.s.w.c.s.StandardServletEnvironment     : Adding [systemProperties] PropertySource with lowest search preced
2017-10-03 08:52:04.425 DEBUG 1 --- [           main] o.s.w.c.s.StandardServletEnvironment     : Adding [systemEnvironment] PropertySource with lowest search prece
2017-10-03 08:52:04.426 DEBUG 1 --- [           main] o.s.w.c.s.StandardServletEnvironment     : Initialized StandardServletEnvironment with PropertySources [servl
ms,systemProperties,systemEnvironment]
2017-10-03 08:52:04.476  INFO 1 --- [           main] com.amishra.RestfulApplication           : Starting RestfulApplication v0.0.1-SNAPSHOT on e99b8bd7d20a with P
ot in /)
2017-10-03 08:52:04.533  INFO 1 --- [           main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationCon
: startup date [Tue Oct 03 08:52:04 UTC 2017]; root of context hierarchy
2017-10-03 08:52:04.661 DEBUG 1 --- [           main] o.s.w.c.s.StandardServletEnvironment     : Adding [class path resource [application.properties]] PropertySour
2017-10-03 08:52:05.466  INFO 1 --- [           main] o.s.b.f.s.DefaultListableBeanFactory     : Overriding bean definition for bean 'beanNameViewResolver': replac
tract=false; lazyInit=false; autowireMode=3; dependencyCheck=0; autowireCandidate=true; primary=false; factoryBeanName=org.springframework.boot.autoconfigure.web.E
iewConfiguration; factoryMethodName=beanNameViewResolver; initMethodName=null; destroyMethodName=(inferred); defined in class path resource [org/springframework/bo
ation$WhitelabelErrorViewConfiguration.class]] with [Root bean: class [null]; scope=; abstract=false; lazyInit=false; autowireMode=3; dependencyCheck=0; autowireCa
ame=org.springframework.boot.autoconfigure.web.WebMvcAutoConfiguration$WebMvcAutoConfigurationAdapter; factoryMethodName=beanNameViewResolver; initMethodName=null;
```

docker ps



```
[[cmperoot@centos7desktop hw2]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
655c6509b9ba        app_akshay_673      "java -Dmongodb.ho..."   4 seconds ago       Up 2 seconds        0.0.0.0:8080->8080/tcp   appAkshay673
1856fae467dc        mongo               "docker-entrypoint..."   10 minutes ago      Up 10 minutes       27017/tcp                dbAkshay673
[cmperoot@centos7desktop hw2]$ 
```

Steps to deploy Web Server to Containers:

It is not required to deploy web server separately to containers and then deploy war file, web app is completely packaged as a JAR file and it has an embedded tomcat web server already. When we build docker image, it happens automatically through the jar in which tomcat is embedded.

Modifications in the REST Server Backend Code

Not needed, because we have multiple externally configurable properties in the backend code already.

Known issues: None

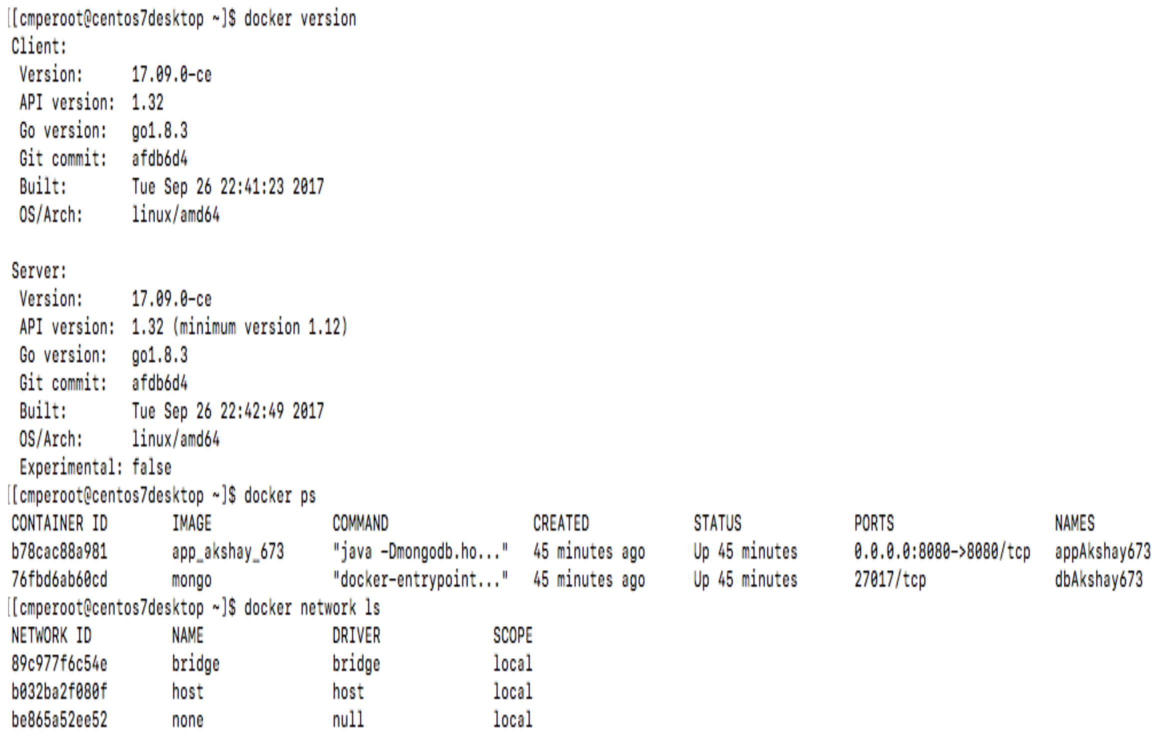
Additional features you are proud of:
We can use default application properties or simply the ones we obtain by linking containers easily. The backend code has been designed in a generic way to achieve this flexibility and application of Docker leverages this to its best use here.

Q4. While both containers are running on host1, include the screenshots of the followings on host1

Answer:

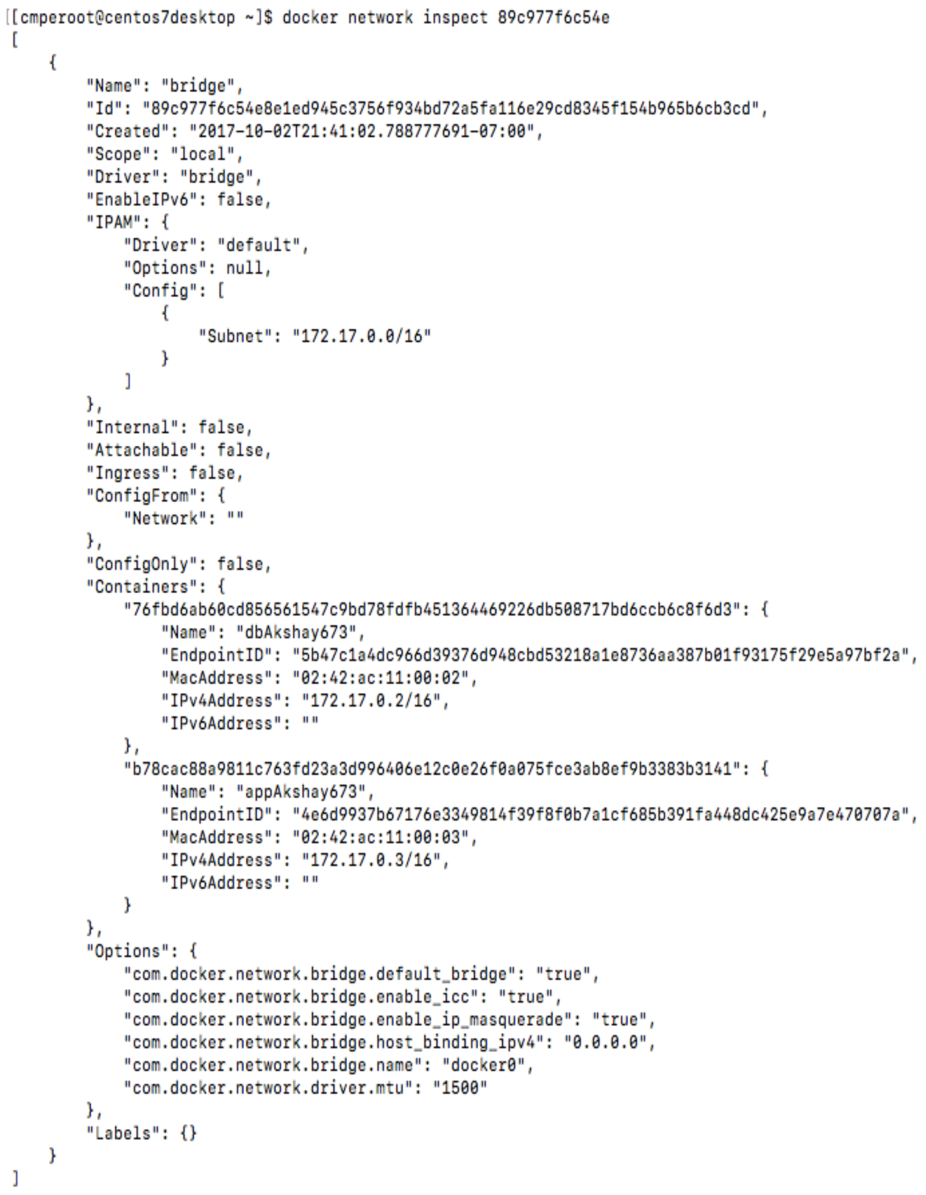
Host 1:

docker version & docker ps



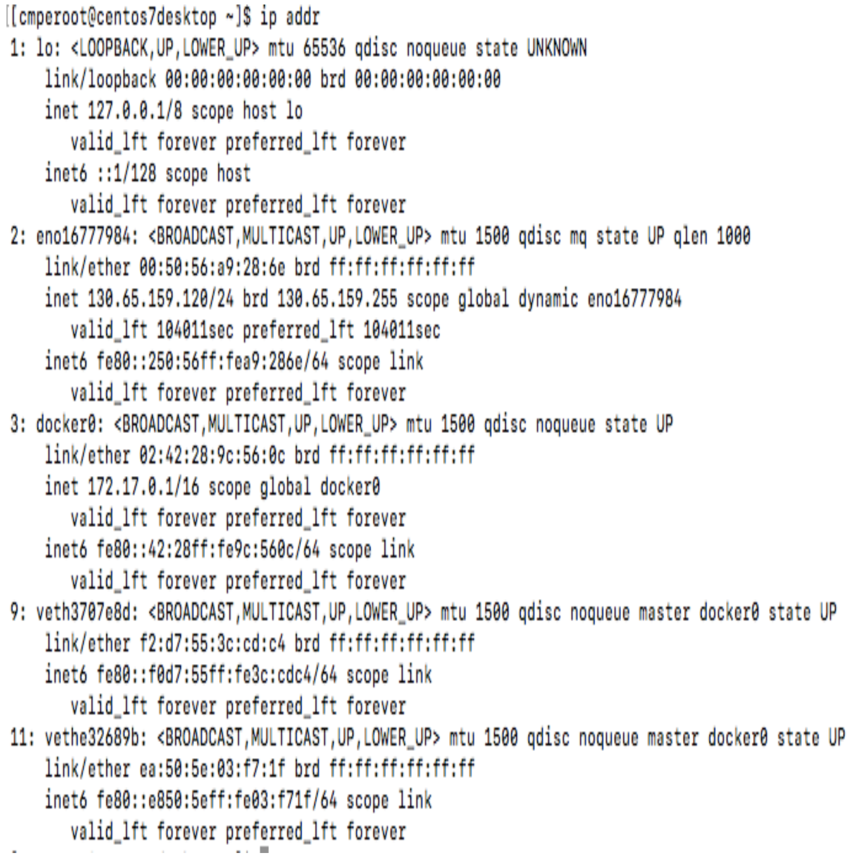
```
[cmperoot@centos7desktop ~]$ docker version
Client:
 Version:      17.09.0-ce
 API version:  1.32
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:41:23 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.09.0-ce
 API version:  1.32 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:42:49 2017
 OS/Arch:      linux/amd64
 Experimental: false
[cmperoot@centos7desktop ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
b78cac88a981        app_akshay_673      "java -Dmongodb.ho..."   45 minutes ago      Up 45 minutes       0.0.0.0:8080->8080/tcp   appAkshay673
76fbd6ab60cd        mongo               "docker-entrypoint..."   45 minutes ago      Up 45 minutes       27017/tcp                dbAkshay673
[cmperoot@centos7desktop ~]$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
89c977f6c54e        bridge              bridge              local
b032ba2f080f        host                host                local
be865a52ee52        none                null                local
```

docker network inspect



```
[[cmperoot@centos7desktop ~]$ docker network inspect 89c977f6c54e
[
    {
        "Name": "bridge",
        "Id": "89c977f6c54e8e1ed945c3756f934bd72a5fa116e29cd8345f154b965b6cb3cd",
        "Created": "2017-10-02T21:41:02.788777691-07:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "76fbd6ab60cd856561547c9bd78fdfb451364469226db508717bd6ccb6c8f6d3": {
                "Name": "dbAkshay673",
                "EndpointID": "5b47c1a4dc966d39376d948cbd53218a1e8736aa387b01f93175f29e5a97bf2a",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "b78cac88a9811c763fd23a3d996406e12c0e26f0a075fce3ab8ef9b3383b3141": {
                "Name": "appAkshay673",
                "EndpointID": "4e6d9937b67176e3349814f39f8f0b7a1cf685b391fa448dc425e9a7e470707a",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

ip addr

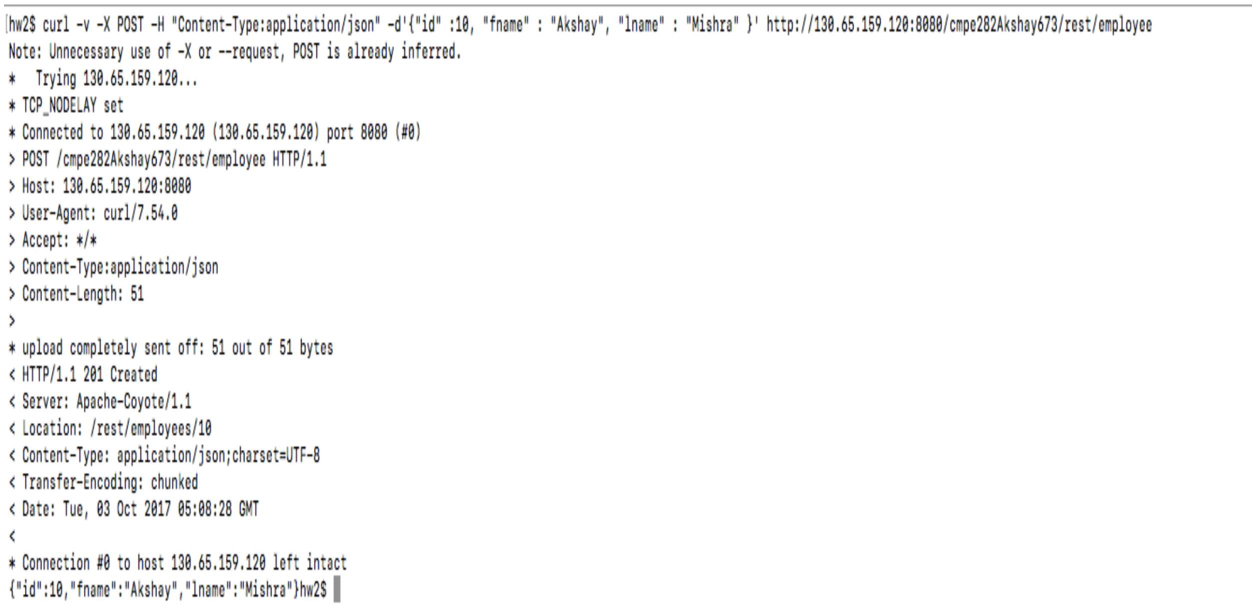


```
[[cmperoot@centos7desktop ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno16777984: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:50:56:a9:28:6e brd ff:ff:ff:ff:ff:ff
    inet 130.65.159.120/24 brd 130.65.159.255 scope global dynamic eno16777984
       valid_lft 104011sec preferred_lft 104011sec
    inet6 fe80::250:56ff:fea9:286e/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:28:9c:56:0c brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:28ff:fe9c:560c/64 scope link
       valid_lft forever preferred_lft forever
9: veth3707e8d: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP
    link/ether f2:d7:55:3c:cd:c4 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::f0d7:55ff:fe3c:cdc4/64 scope link
       valid_lft forever preferred_lft forever
11: vethe32689b: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP
    link/ether ea:50:5e:03:f7:1f brd ff:ff:ff:ff:ff:ff
    inet6 fe80::e850:5eff:fe03:f71f/64 scope link
       valid_lft forever preferred_lft forever
```

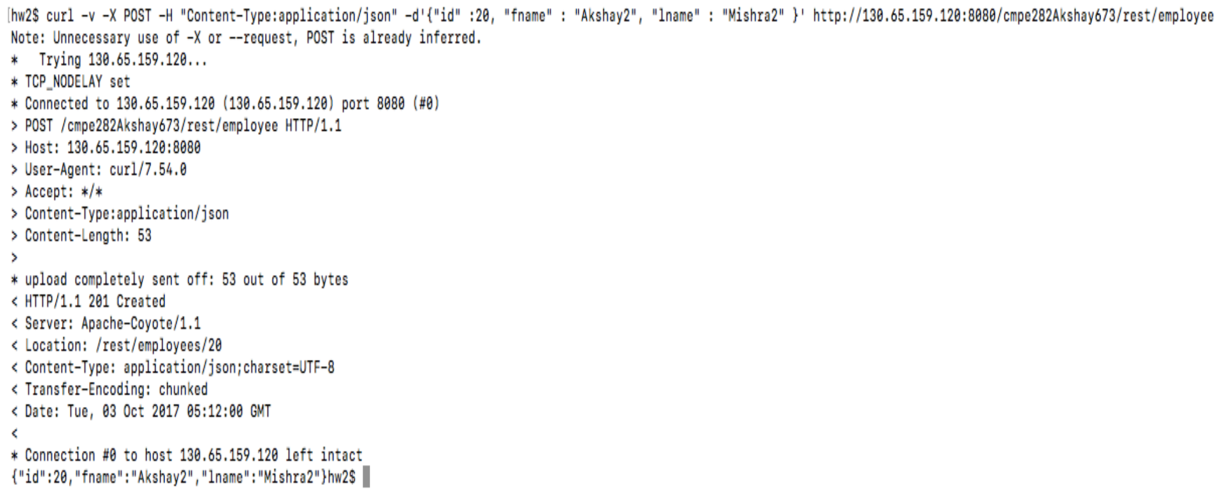
Q5. On host2, use REST client to issue the following requests and include screenshots of REST request and response (method, URL, HTTP headers) - success cases only:

Answer:

POST for Employee 10, 20

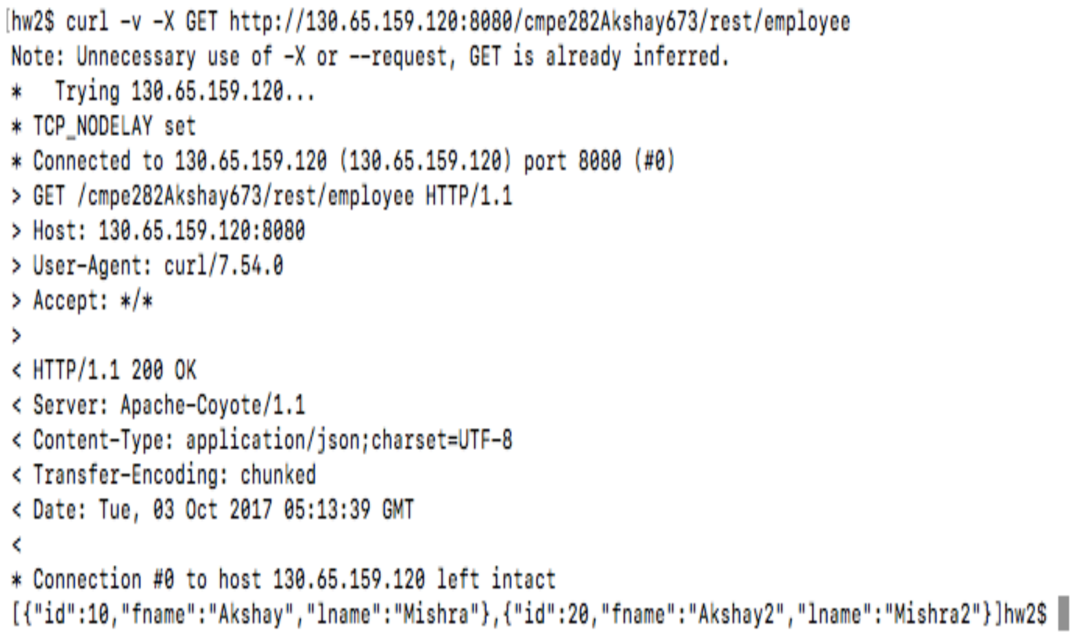


```
hw2$ curl -v -X POST -H "Content-Type:application/json" -d'{"id" :10, "fname" : "Akshay", "lname" : "Mishra" }' http://130.65.159.120:8080/cmpe282Akshay673/rest/employee
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> POST /cmpe282Akshay673/rest/employee HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type:application/json
> Content-Length: 51
>
* upload completely sent off: 51 out of 51 bytes
< HTTP/1.1 201 Created
< Server: Apache-Coyote/1.1
< Location: /rest/employees/10
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:08:28 GMT
<
* Connection #0 to host 130.65.159.120 left intact
{"id":10,"fname":"Akshay","lname":"Mishra"}hw2$
```



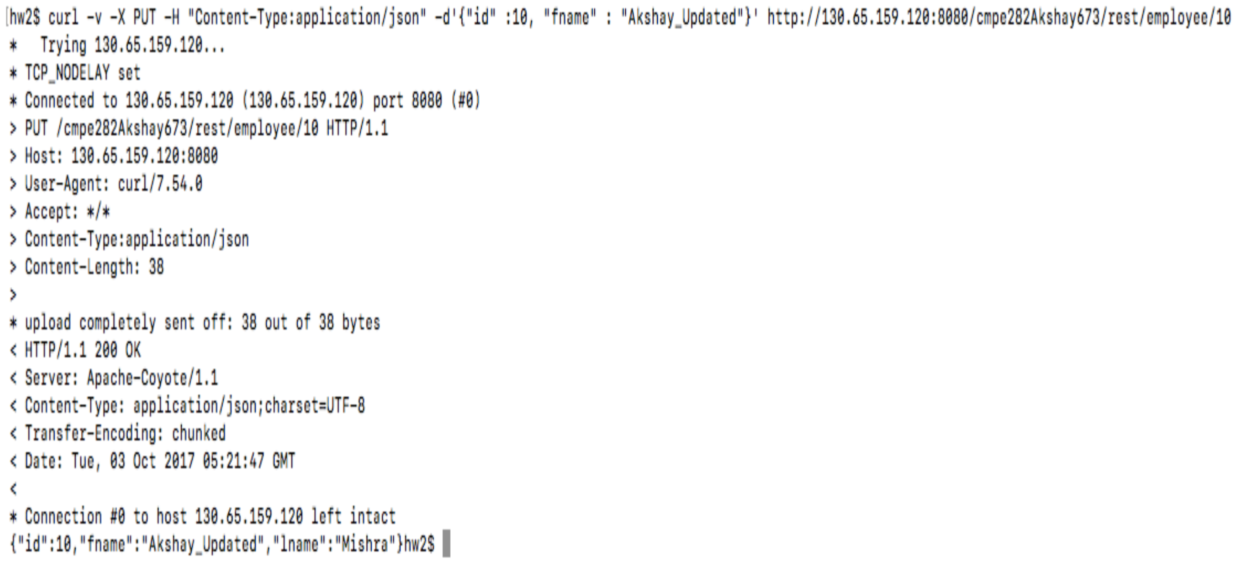
```
[hw2$ curl -v -X POST -H "Content-Type:application/json" -d'{"id" :20, "fname" : "Akshay2", "lname" : "Mishra2" }' http://130.65.159.120:8080/cmpe282Akshay673/rest/employee
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> POST /cmpe282Akshay673/rest/employee HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type:application/json
> Content-Length: 53
>
* upload completely sent off: 53 out of 53 bytes
< HTTP/1.1 201 Created
< Server: Apache-Coyote/1.1
< Location: /rest/employees/20
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:12:00 GMT
<
* Connection #0 to host 130.65.159.120 left intact
{"id":20,"fname":"Akshay2","lname":"Mishra2"}hw2$
```

GET All Employees



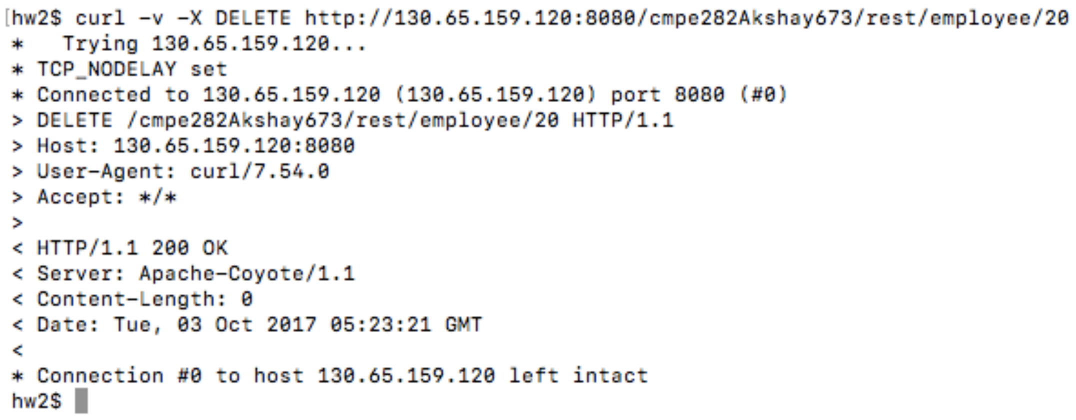
```
[hw2$ curl -v -X GET http://130.65.159.120:8080/cmpe282Akshay673/rest/employee
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> GET /cmpe282Akshay673/rest/employee HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:13:39 GMT
<
* Connection #0 to host 130.65.159.120 left intact
[{"id":10,"fname":"Akshay","lname":"Mishra"},{"id":20,"fname":"Akshay2","lname":"Mishra2"}]hw2$
```

PUT Employee 10 only fname



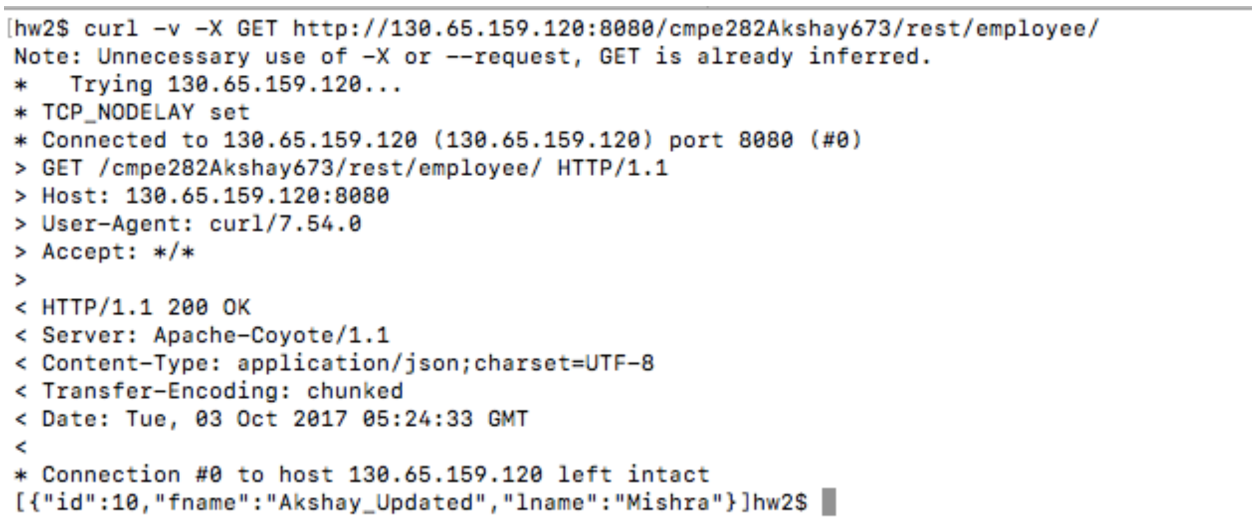
```
[hw2$ curl -v -X PUT -H "Content-Type:application/json" -d'{"id" :10, "fname" : "Akshay_Updated"}' http://130.65.159.120:8080/cmpe282Akshay673/rest/employee/10
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> PUT /cmpe282Akshay673/rest/employee/10 HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type:application/json
> Content-Length: 38
>
* upload completely sent off: 38 out of 38 bytes
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:21:47 GMT
<
* Connection #0 to host 130.65.159.120 left intact
{"id":10,"fname":"Akshay_Updated","lname":"Mishra"}hw2$
```

DELETE Employee 20



```
[hw2$ curl -v -X DELETE http://130.65.159.120:8080/cmpe282Akshay673/rest/employee/20
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> DELETE /cmpe282Akshay673/rest/employee/20 HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Length: 0
< Date: Tue, 03 Oct 2017 05:23:21 GMT
<
* Connection #0 to host 130.65.159.120 left intact
hw2$
```

GET All Employees



```
[hw2$ curl -v -X GET http://130.65.159.120:8080/cmpe282Akshay673/rest/employee/
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> GET /cmpe282Akshay673/rest/employee/ HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:24:33 GMT
<
* Connection #0 to host 130.65.159.120 left intact
[{"id":10,"fname":"Akshay_Updated","lname":"Mishra"}]hw2$
```

EXTRA CREDIT

Q6. In addition to the original homework, use docker compose to build and deploy containers in Q2 and Q3.

Answer:

Docker-compose File:

```
version: '2'
services:
  app:
    build: .
    container_name: appAkshay673
    ports:
    - 8080:8080
    depends_on:
    - dbAkshay673
    links:
    - dbAkshay673:dbAkshay673
    restart: always

  dbAkshay673:
    image: mongo
    container_name: dbAkshay673

volumes:
  data:
    driver: local
```

Steps to use docker-compose:

This docker-compose file defines two services namely ‘app’ and ‘dbAkshay673’ with their right configurations. Here we are linking two containers using a default bridge network.

We are specifying dockerfile loication using build option for Web App and image name for MongoDB container.

In order to run the setup using docker-compose, simply run below command from the directory where both the Web App Dockerfile as well as above docker-compose file is stored.

```
Docker-compose up -d --build
```

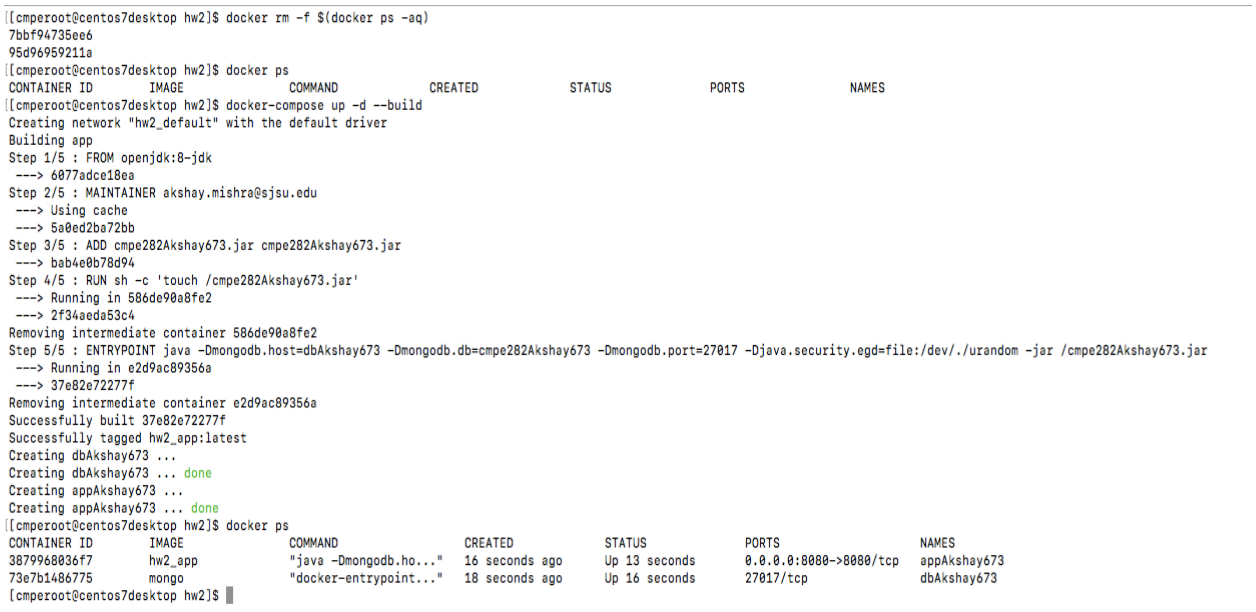
This will basically spin up both the containers with given configurations such as container names, links, port mapping and volume mapping.

Scope for more automation

There is no need to copy any war file manually, the process is already fully automated. At most what one can do is map the volume in which JAR file is kept to a directory inside the container. That way the JAR won’t be needed to copy in the docker-file, and it will be mapped directly from the host file. The latest jar can be updated and container can pick it up from there.

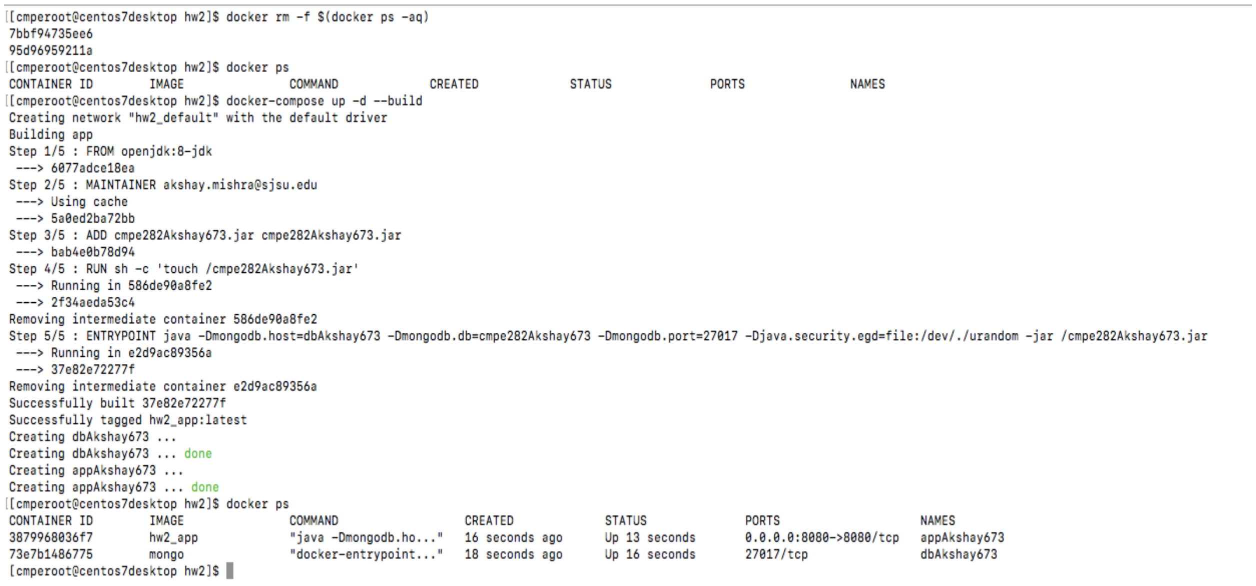
Screenshots:

Docker compose command to spin up both the containers:



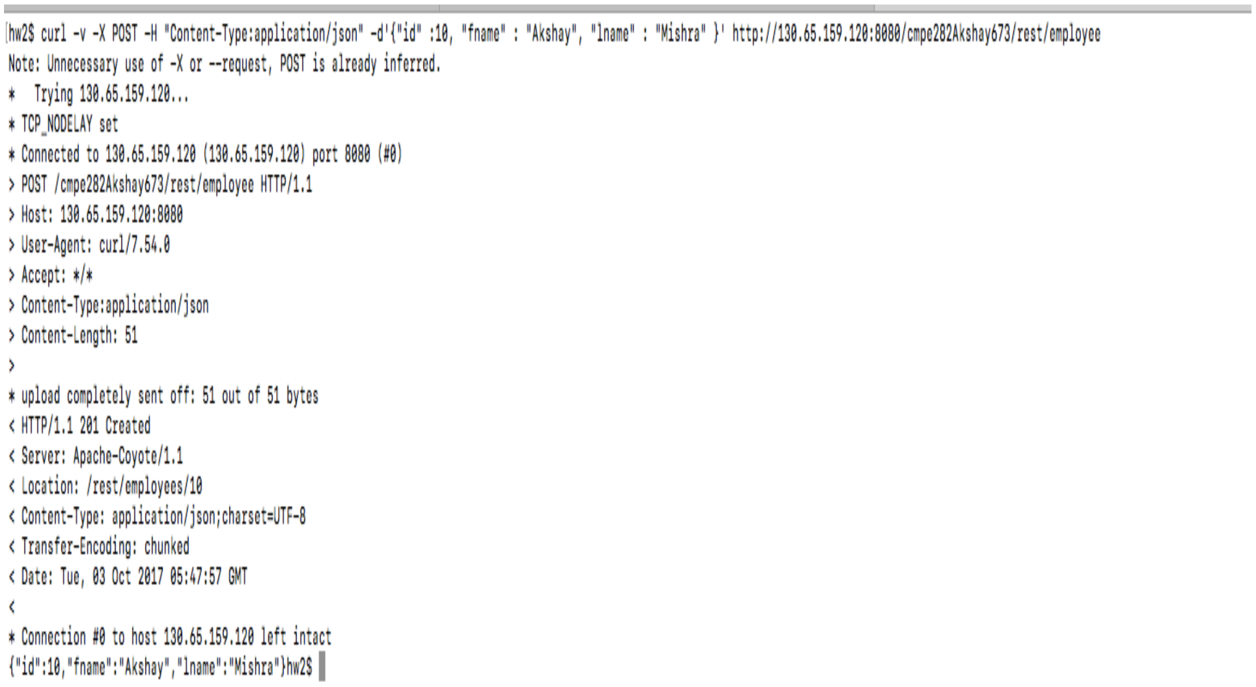
```
[[cmperoot@centos7desktop hw2]$ docker rm -f $(docker ps -aq)
7bbf94735ee6
95d96959211a
[[cmperoot@centos7desktop hw2]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[[cmperoot@centos7desktop hw2]$ docker-compose up -d --build
Creating network "hw2_default" with the default driver
Building app
Step 1/5 : FROM openjdk:8-jdk
 ---> 6077adce18ea
Step 2/5 : MAINTAINER akshay.mishra@sjsu.edu
 ---> Using cache
 ---> 5a0ed2ba72bb
Step 3/5 : ADD cmpe282Akshay673.jar cmpe282Akshay673.jar
 ---> bab4e0b78d94
Step 4/5 : RUN sh -c 'touch /cmpe282Akshay673.jar'
 ---> Running in 586de90a8fe2
 ---> 2f34aeda53c4
Removing intermediate container 586de90a8fe2
Step 5/5 : ENTRYPOINT java -Dmongodb.host=dbAkshay673 -Dmongodb.db=cmpe282Akshay673 -Dmongodb.port=27017 -Djava.security.egd=file:/dev/./urandom -jar /cmpe282Akshay673.jar
 ---> Running in e2d9ac89356a
 ---> 37e82e72277f
Removing intermediate container e2d9ac89356a
Successfully built 37e82e72277f
Successfully tagged hw2_app:latest
Creating dbAkshay673 ...
Creating dbAkshay673 ... done
Creating appAkshay673 ...
Creating appAkshay673 ... done
[[cmperoot@centos7desktop hw2]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
3879968036f7        hw2_app             "java -Dmongodb.ho..."   16 seconds ago      Up 13 seconds       0.0.0.0:8080->8080/tcp   appAkshay673
73e7b1486775        mongo               "docker-entrypoint..."   18 seconds ago      Up 16 seconds       27017/tcp                dbAkshay673
[cmperoot@centos7desktop hw2]$ 
```

docker ps

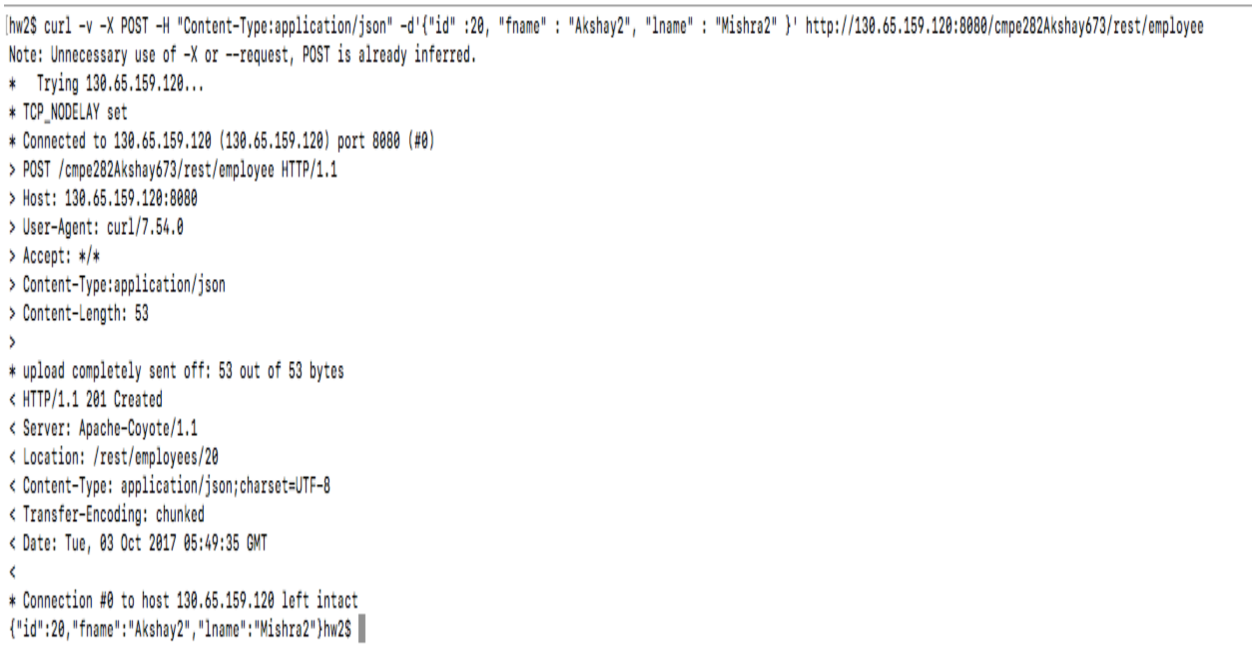


```
[[cmperoot@centos7desktop hw2]$ docker rm -f $(docker ps -aq)
7bbf94735ee6
95d96959211a
[[cmperoot@centos7desktop hw2]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[[cmperoot@centos7desktop hw2]$ docker-compose up -d --build
Creating network "hw2_default" with the default driver
Building app
Step 1/5 : FROM openjdk:8-jdk
 ---> 6077adce18ea
Step 2/5 : MAINTAINER akshay.mishra@sjsu.edu
 ---> Using cache
 ---> 5a0ed2ba72bb
Step 3/5 : ADD cmpe282Akshay673.jar cmpe282Akshay673.jar
 ---> bab4e0b78d94
Step 4/5 : RUN sh -c 'touch /cmpe282Akshay673.jar'
 ---> Running in 586de90a8fe2
 ---> 2f34aeda53c4
Removing intermediate container 586de90a8fe2
Step 5/5 : ENTRYPOINT java -Dmongodb.host=dbAkshay673 -Dmongodb.db=cmpe282Akshay673 -Dmongodb.port=27017 -Djava.security.egd=file:/dev/./urandom -jar /cmpe282Akshay673.jar
 ---> Running in e2d9ac89356a
 ---> 37e82e72277f
Removing intermediate container e2d9ac89356a
Successfully built 37e82e72277f
Successfully tagged hw2_app:latest
Creating dbAkshay673 ...
Creating dbAkshay673 ... done
Creating appAkshay673 ...
Creating appAkshay673 ... done
[[cmperoot@centos7desktop hw2]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
3879968036f7        hw2_app             "java -Dmongodb.ho..."   16 seconds ago      Up 13 seconds       0.0.0.0:8080->8080/tcp   appAkshay673
73e7b1486775        mongo               "docker-entrypoint..."   18 seconds ago      Up 16 seconds       27017/tcp                dbAkshay673
[cmperoot@centos7desktop hw2]$ 
```

First two cases with docker-compose:

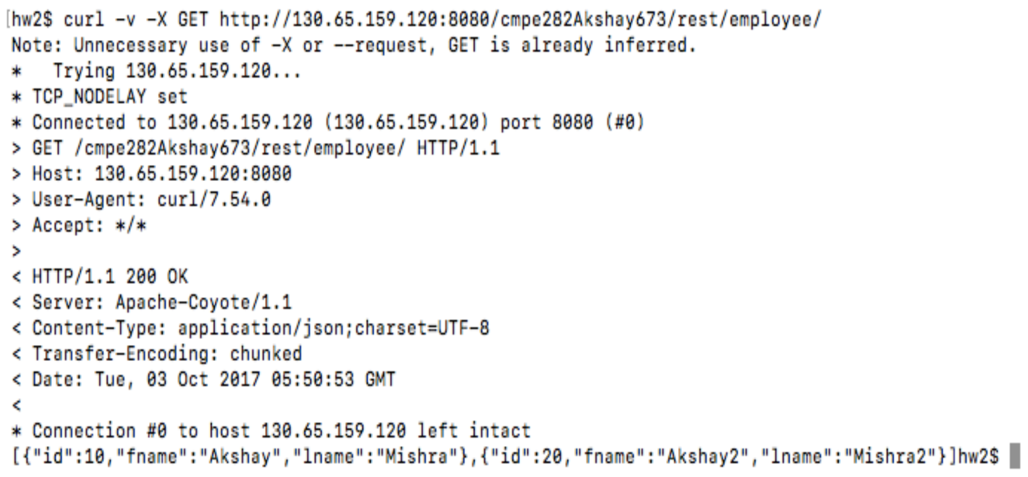


```
[hw2$ curl -v -X POST -H "Content-Type:application/json" -d'{"id" :10, "fname" : "Akshay", "lname" : "Mishra" }' http://130.65.159.120:8080/cmpe282Akshay673/rest/employee
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> POST /cmpe282Akshay673/rest/employee HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type:application/json
> Content-Length: 51
>
* upload completely sent off: 51 out of 51 bytes
< HTTP/1.1 201 Created
< Server: Apache-Coyote/1.1
< Location: /rest/employees/10
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:47:57 GMT
<
* Connection #0 to host 130.65.159.120 left intact
{"id":10,"fname":"Akshay","lname":"Mishra"}hw2$ 
```



```
hw2$ curl -v -X POST -H "Content-Type:application/json" -d'{"id" :20, "fname" : "Akshay2", "lname" : "Mishra2" }' http://130.65.159.120:8080/cmpe282Akshay673/rest/employee
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> POST /cmpe282Akshay673/rest/employee HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Type:application/json
> Content-Length: 53
>
* upload completely sent off: 53 out of 53 bytes
< HTTP/1.1 201 Created
< Server: Apache-Coyote/1.1
< Location: /rest/employees/20
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:49:35 GMT
<
* Connection #0 to host 130.65.159.120 left intact
{"id":20,"fname":"Akshay2","lname":"Mishra2"}hw2$
```

Get All Employees



```
hw2$ curl -v -X GET http://130.65.159.120:8080/cmpe282Akshay673/rest/employee/
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 130.65.159.120...
* TCP_NODELAY set
* Connected to 130.65.159.120 (130.65.159.120) port 8080 (#0)
> GET /cmpe282Akshay673/rest/employee/ HTTP/1.1
> Host: 130.65.159.120:8080
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Tue, 03 Oct 2017 05:50:53 GMT
<
* Connection #0 to host 130.65.159.120 left intact
[{"id":10,"fname":"Akshay","lname":"Mishra"},{"id":20,"fname":"Akshay2","lname":"Mishra2"}]hw2$
```