1. In the previous exercise a container was created based on image docker/whalesay image as follows.

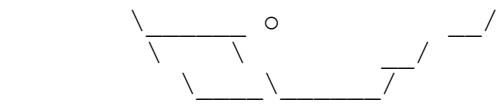
```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container run docker/whalesay
cowsay "An Important Message"

 _________________________
< An Important Message >
 -------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \     \        __/
          \____\______/
```

Redo this activity using Dockerfile. Create a folder "whale" and a file "Dockerfile" inside with following content.

```
osgdev@TG-DevOps-OS004:~/dockerlab/whale$ cat Dockerfile
FROM docker/whalesay:latest
CMD echo "An Important Message" | cowsay
```

Build a new image with the above Dockerfile which will automatically print the same message each time.

```
osgdev@TG-DevOps-OS004:~/dockerlab/whale$ docker build -t whale001 .
Sending build context to Docker daemon  391.2kB
Step 1/2 : FROM docker/whalesay:latest
 ---> 6b362a9f73eb
Step 2/2 : CMD echo "An Important Message" | cowsay
 ---> Running in c5a06f7ce0c0
Removing intermediate container c5a06f7ce0c0
 ---> 706bfd241e32
Successfully built 706bfd241e32
Successfully tagged whale001:latest

osgdev@TG-DevOps-OS004:~/dockerlab/whale$ docker container run whale001

 _________________________
< An Important Message >
 -------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
```



---

2. Creating a simple Java Application in Docker container

Create a simple jar file to say Hello World and containerize the same.

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ vi HelloWorld.java

osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat HelloWorld.java
public class HelloWorld {

  public static void main(String[] args){

    System.out.println("Hello World :) ");

  }
}

osgdev@TG-DevOps-OS004:~/dockerlab/java$ javac HelloWorld.java

osgdev@TG-DevOps-OS004:~/dockerlab/java$ java HelloWorld
Hello World :)

osgdev@TG-DevOps-OS004:~/dockerlab/java$ ls
HelloWorld.class  HelloWorld.java

osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat manifest.txt
Manifest-Version: 1.0
Created-By: training
Main-Class: HelloWorld

osgdev@TG-DevOps-OS004:~/dockerlab/java$ jar cfm HelloWorld.jar
manifest.txt HelloWorld.class

osgdev@TG-DevOps-OS004:~/dockerlab/java$ ls
HelloWorld.class  HelloWorld.jar  HelloWorld.java  manifest.txt

osgdev@TG-DevOps-OS004:~/dockerlab/java$ java -jar HelloWorld.jar
Hello World :)
```

Add Dockerfile to display the same.

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8
CMD java -jar HelloWorld.jar

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava .
```

```
Sending build context to Docker daemon   7.68kB
Step 1/2 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/2 : CMD java -jar HelloWorld.jar
 ---> Running in 7291d4d592fa
Removing intermediate container 7291d4d592fa
 ---> ded56001d590
Successfully built ded56001d590
Successfully tagged hellojava:latest


osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run -it
hellojava
Error: Unable to access jarfile HelloWorld.jar

Issue can be resolved by adding HelloWorld.jar to image. Note when we
used '.' current folder, that not only specify availability of
Dockerfile, but also indicates that image can use any file available in
this work folder '.'  Hence HelloWorld.jar available in '.' folder is
added to image.

osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8

ADD HelloWorld.jar HelloWorld.jar

CMD java -jar HelloWorld.jar

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava1 .
Sending build context to Docker daemon   7.68kB
Step 1/3 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/3 : ADD HelloWorld.jar HelloWorld.jar
 ---> 148064282946
Step 3/3 : CMD java -jar HelloWorld.jar
 ---> Running in d3b3029cba4b
Removing intermediate container d3b3029cba4b
 ---> 1a1bdaed874f
Successfully built 1a1bdaed874f
Successfully tagged hellojava1:latest

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run hellojava1
Hello World :)
```

Check inside the container. To check content inside the container, overrule the command set in the image "CMD java -jar HelloWorld.jar" using a command externally with docker container run command as we are using "/bin/bash" here.

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run -it
hellojava1 /bin/bash
root@32405ddd62b9:/# pwd
/
root@32405ddd62b9:/# ls
```

```
HelloWorld.jar   boot  etc   lib    media  opt      root  sbin  sys  usr
bin         dev   home  lib64  mnt   proc      run   srv   tmp  var
root@32405ddd62b9:/# exit
exit
```

Note the availability of HelloWorld.jar file under '/' folder by default.

Instead of allowing '/' to be the default working directory, you can set a different working directory.

Modify the Dockerfile as below.

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ vi Dockerfile

osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8

WORKDIR /appjava

ADD HelloWorld.jar HelloWorld.jar

CMD java -jar HelloWorld.jar

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava2 .
Sending build context to Docker daemon   7.68kB
Step 1/4 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/4 : WORKDIR /appjava
 ---> Using cache
 ---> 8e8008c6d6ac
Step 3/4 : ADD HelloWorld.jar HelloWorld.jar
 ---> 7788a94b1e08
Step 4/4 : CMD java -jar HelloWorld.jar
 ---> Running in 2bc66638cb0b
Removing intermediate container 2bc66638cb0b
 ---> 136e0672b728
Successfully built 136e0672b728
Successfully tagged hellojava2:latest
osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run hellojava2
Hello World :)
osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run -it
hellojava2 /bin/bash
root@d91c694a5611:/appjava# pwd
/appjava
root@d91c694a5611:/appjava# ls
HelloWorld.jar
root@d91c694a5611:/appjava# exit
exit
```

Add the commands "javac HelloWorld.java" and "jar cfm HelloWorld.jar manifest.txt HelloWorld.class" inside Dockerfile.

Create Dockerfile as follows:

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8
WORKDIR /appjava
COPY . /appjava

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava3 .
Sending build context to Docker daemon   7.68kB
Step 1/3 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/3 : WORKDIR /appjava
 ---> Using cache
 ---> 8e8008c6d6ac
Step 3/3 : COPY . /appjava
 ---> cc1704b1bf24
Successfully built cc1704b1bf24
Successfully tagged hellojava3:latest

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run -it
hellojava3 /bin/bash
root@f69e084050c8:/appjava# ls
Dockerfile HelloWorld.java manifest.txt
root@f69e084050c8:/appjava# exit
exit

osgdev@TG-DevOps-OS004:~/dockerlab/java$ ls
Dockerfile HelloWorld.java  manifest.txt
```

Add the compilation command "javac HelloWorld.java"

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8
WORKDIR /appjava
COPY . /appjava
RUN javac HelloWorld.java

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava4 .
<<  Output not shown  >>

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run -it
hellojava4 /bin/bash
root@61c8fc88025e:/appjava# ls
Dockerfile  HelloWorld.class  HelloWorld.java  df  manifest.txt
root@61c8fc88025e:/appjava# exit
exit

osgdev@TG-DevOps-OS004:~/dockerlab/java$ ls
Dockerfile  HelloWorld.java  manifest.txt
```

Note: Observe the change is happening only in the image reflected in container.

Create jar file using the command "jar cfm HelloWorld.jar manifest.txt HelloWorld.class"

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8
WORKDIR /appjava
COPY . /appjava
RUN javac HelloWorld.java
RUN jar cfm HelloWorld.jar manifest.txt HelloWorld.class

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava4 .
<<  Output not shown  >>

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run -it
hellojava5 /bin/bash
root@581bbbe0edf4:/appjava# ls
Dockerfile  HelloWorld.class  HelloWorld.jar  HelloWorld.java  df
manifest.txt
root@581bbbe0edf4:/appjava# exit
exit
```

Finally set "java -jar HelloWorld.jar" as the command CMD to be executed by default whenever container run based on this image

```
osgdev@TG-DevOps-OS004:~/dockerlab/java$ cat Dockerfile
FROM java:8
WORKDIR /appjava
COPY . /appjava
RUN javac HelloWorld.java
RUN jar cfm HelloWorld.jar manifest.txt HelloWorld.class
CMD java -jar HelloWorld.jar

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker build -t hellojava6 .
Sending build context to Docker daemon   5.12kB
Step 1/6 : FROM java:8
 ---> d23bdf5b1b1b
Step 2/6 : WORKDIR /appjava
 ---> Using cache
 ---> 8e8008c6d6ac
Step 3/6 : COPY . /appjava
 ---> ab3a849e71d7
Step 4/6 : RUN javac HelloWorld.java
 ---> Running in 15e0ea8d81a1
Removing intermediate container 15e0ea8d81a1
 ---> 7b0fac9e364b
Step 5/6 : RUN jar cfm HelloWorld.jar manifest.txt HelloWorld.class
 ---> Running in 8ddac6f4cb34
Removing intermediate container 8ddac6f4cb34
 ---> 4aaf269031af
Step 6/6 : CMD java -jar HelloWorld.jar
 ---> Running in b771eacc021e
Removing intermediate container b771eacc021e
 ---> 0d9a1c04714b
Successfully built 0d9a1c04714b
Successfully tagged hellojava6:latest

osgdev@TG-DevOps-OS004:~/dockerlab/java$ docker container run hellojava6
```

```
Hello World :)
```

---

3. Create a Python APP.

Create following three files with specified contents inside python folder:

```
osgdev@TG-DevOps-OS004:~/dockerlab/python$ vi app.py
osgdev@TG-DevOps-OS004:~/dockerlab/python$ cat app.py
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2,
socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
           "<b>Hostname:</b> {hostname}<br/>" \
           "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"),
hostname=socket.gethostname(), visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)

osgdev@TG-DevOps-OS004:~/dockerlab/python$ vi requirements.txt
osgdev@TG-DevOps-OS004:~/dockerlab/python$ cat requirements.txt
Flask
Redis

osgdev@TG-DevOps-OS004:~/dockerlab/python$ vi Dockerfile
osgdev@TG-DevOps-OS004:~/dockerlab/python$ cat Dockerfile
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app
```

```
# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Build the image: (If a docker image is not available the following operation should automatically pull the image. However it failed in first attemp, and hence needed to pull the required image separately.

```
osgdev@TG-DevOps-OS004:~/dockerlab/python$ docker build -t sayhello .
Sending build context to Docker daemon  4.608kB
Step 1/7 : FROM python:2.7-slim
2.7-slim: Pulling from library/python
Get https://registry-
1.docker.io/v2/library/python/manifests/sha256:62e7058e99f6e17db9b6515dc3
f655089c7a3949ec8de1c4f5cd7257cb3dead8: read tcp 10.199.0.104:38186-
>35.169.231.249:443: read: connection reset by peer

osgdev@TG-DevOps-OS004:~/dockerlab/python$ docker image pull python:2.7-
slim
2.7-slim: Pulling from library/python
b0568b191983: Pull complete
55a7da9473ae: Pull complete
422d2e7f1272: Pull complete
8fb86f1cff1c: Pull complete
Digest:
sha256:5761a68d4162c7ed555a4f21a09616a5f9f5c4072a1742ad7354c9c7fdd726a2
Status: Downloaded newer image for python:2.7-slim

osgdev@TG-DevOps-OS004:~/dockerlab/python$ docker build -t sayhello .
Sending build context to Docker daemon  4.608kB
Step 1/7 : FROM python:2.7-slim
 ---> b16fde09c92c
Step 2/7 : WORKDIR /app
Removing intermediate container 01a4697e2067
 ---> 77fae6c6b4a3
Step 3/7 : ADD . /app
 ---> dbb42afaf5be
Step 4/7 : RUN pip install --trusted-host pypi.python.org -r
requirements.txt
 ---> Running in 0ce470c55b2e
Collecting Flask (from -r requirements.txt (line 1))
```

```
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting Redis (from -r requirements.txt (line 2))
  Downloading redis-2.10.6-py2.py3-none-any.whl (64kB)
Collecting itsdangerous>=0.21 (from Flask->-r requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2>=2.4 (from Flask->-r requirements.txt (line 1))
  Downloading Jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting Werkzeug>=0.7 (from Flask->-r requirements.txt (line 1))
  Downloading Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
Collecting click>=2.0 (from Flask->-r requirements.txt (line 1))
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->Flask->-r requirements.txt
(line 1))
  Downloading MarkupSafe-1.0.tar.gz
Building wheels for collected packages: itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for itsdangerous: started
  Running setup.py bdist_wheel for itsdangerous: finished with status
'done'
  Stored in directory:
/root/.cache/pip/wheels/fc/a8/66/24d655233c757e178d45dea2de22a04c6d92766a
bfb741129a
  Running setup.py bdist_wheel for MarkupSafe: started
  Running setup.py bdist_wheel for MarkupSafe: finished with status
'done'
  Stored in directory:
/root/.cache/pip/wheels/88/a7/30/e39a54a87bcbe25308fa3ca64e8ddc75d9b3e5af
a21ee32d57
Successfully built itsdangerous MarkupSafe
Installing collected packages: itsdangerous, MarkupSafe, Jinja2,
Werkzeug, click, Flask, Redis
Successfully installed Flask-0.12.2 Jinja2-2.10 MarkupSafe-1.0 Redis-
2.10.6 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
You are using pip version 9.0.3, however version 10.0.0 is available.
You should consider upgrading via the 'pip install --upgrade pip'
command.
Removing intermediate container 0ce470c55b2e
 ---> c778a0e26225
Step 5/7 : EXPOSE 80
 ---> Running in 2b27f475c76f
Removing intermediate container 2b27f475c76f
 ---> 572567a6b926
Step 6/7 : ENV NAME World
 ---> Running in bcd583ccf10e
Removing intermediate container bcd583ccf10e
 ---> ada1bb93a487
Step 7/7 : CMD ["python", "app.py"]
 ---> Running in 309977a2cdb0
Removing intermediate container 309977a2cdb0
 ---> 02b691805eb2
Successfully built 02b691805eb2
Successfully tagged sayhello:latest
```

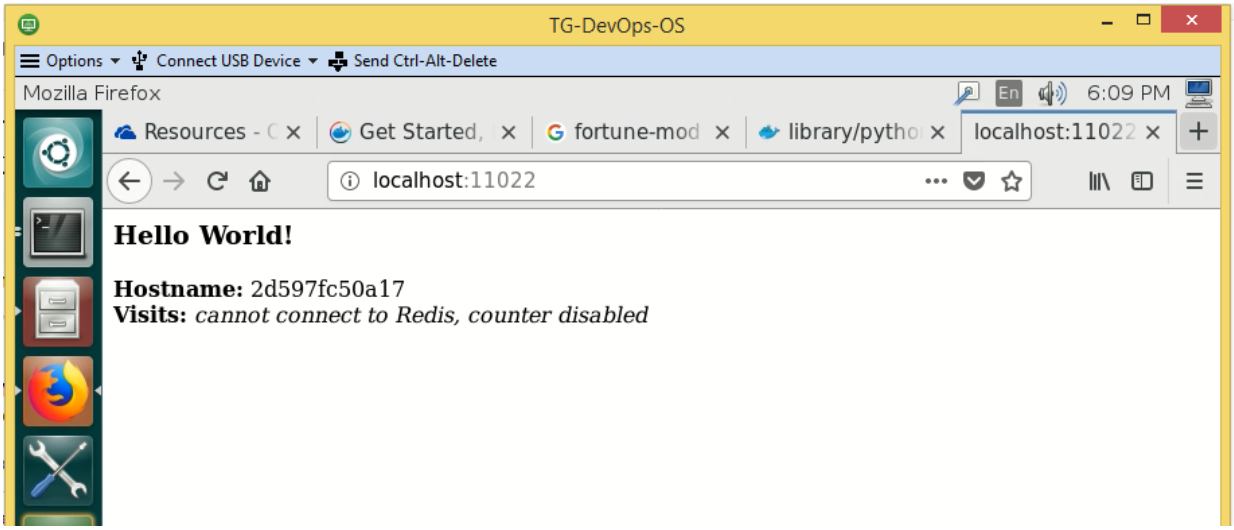
That created a new image called sayhello:latest

```
osgdev@TG-DevOps-OS004:~/dockerlab/python$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
sayhello            latest              02b691805eb2        10 seconds
ago      150MB
```

Let us launch a container with this image:

```
osgdev@TG-DevOps-OS004:~/dockerlab/python$ docker container run -d -p
11022:80 sayhello
7091198300b4db12fe406e6958a60c16abc135056ddb75d9a83ee62fa4767238

osgdev@TG-DevOps-OS004:~/dockerlab/python$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS                   NAMES
7091198300b4        sayhello            "python app.py"     7 seconds ago
Up 6 seconds        0.0.0.0:11022->80/tcp   upbeat_clarke
```

As container is running, you can watch the result on Firefox browser.



---

4. We did containerized an available .war file in the previous exercise as follows.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ ls web
NewApp1  NewApp1.war

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container run -d -p 11055:8080
-v /home/osgdev/dockerlab/web:/usr/local/tomcat/webapps tomcat:8
4a4fe43eeca7548bb194c23a85d76bf2dec9ef810d6ad2a87f69b9fbe8b3359b

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS                     NAMES
4a4fe43eeca7        tomcat:8            "catalina.sh run"   5 seconds ago
Up 4 seconds        0.0.0.0:11055->8080/tcp   elastic_villani
```

You can now create a Dockerfile to work with this.

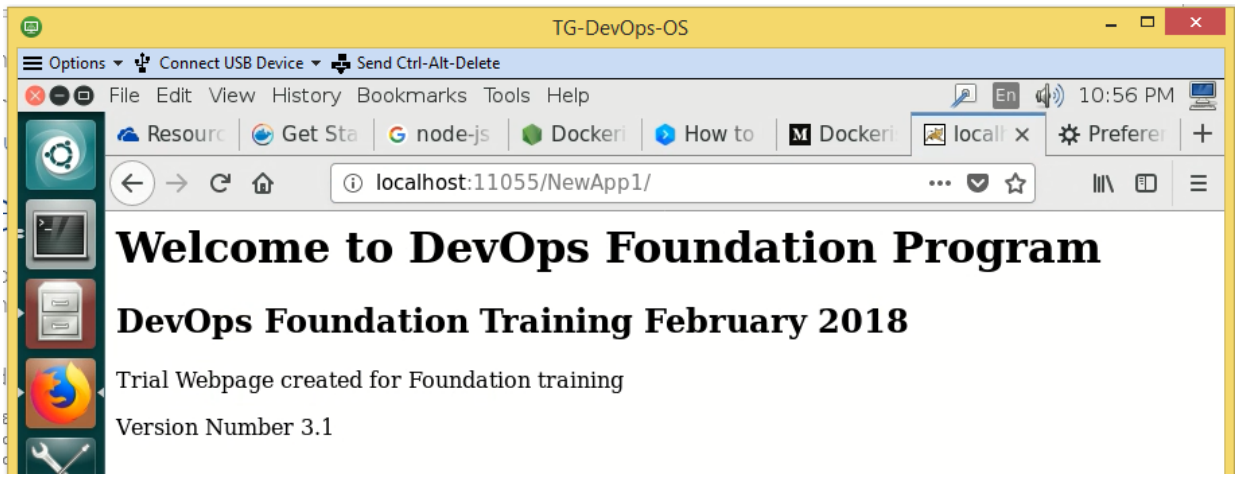
```
osgdev@TG-DevOps-OS004:~/dockerlab/web$ cat Dockerfile
FROM tomcat:8
ADD NewApp1.war /usr/local/tomcat/webapps/
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

You may face an error when you try to build the image due to availability of folder by name "NewApp1" with different permissions.

```
osgdev@TG-DevOps-OS004:~/dockerlab/web$ docker build -t tomcat1 .
error checking context: 'can't stat
'/home/osgdev/dockerlab/web/NewApp1''.

osgdev@TG-DevOps-OS004:~/dockerlab/web$ ls
Dockerfile  NewApp1  NewApp1.war
osgdev@TG-DevOps-OS004:~/dockerlab/web$ vi Dockerfile
osgdev@TG-DevOps-OS004:~/dockerlab/web$ rm -rf NewApp1
rm: cannot remove 'NewApp1': Permission denied
osgdev@TG-DevOps-OS004:~/dockerlab/web$ sudo rm -rf NewApp1
[sudo] password for osgdev:
osgdev@TG-DevOps-OS004:~/dockerlab/web$ ls
Dockerfile  NewApp1.war


osgdev@TG-DevOps-OS004:~/dockerlab/web$ docker build -t tomcat1 .
Sending build context to Docker daemon   5.12kB
Step 1/4 : FROM tomcat:8
 ---> 4db09019de0d
Step 2/4 : ADD NewApp1.war /usr/local/tomcat/webapps/
 ---> 4ca7f2e74a7a
Step 3/4 : EXPOSE 8080
 ---> Running in 8d07f5685777
Removing intermediate container 8d07f5685777
 ---> 1d48c619af91
Step 4/4 : CMD ["catalina.sh", "run"]
 ---> Running in 195b93217a07
Removing intermediate container 195b93217a07
 ---> 29abadd3ae4c
Successfully built 29abadd3ae4c
Successfully tagged tomcat1:latest

osgdev@TG-DevOps-OS004:~/dockerlab/web$ docker run -d -p 11055:8080
tomcat1
a39bcbfac0e001222b5fb29254e60c00c4dfe8f7786f0ae09ba11da3e03103f6

osgdev@TG-DevOps-OS004:~/dockerlab/web$ docker container ls
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS                     NAMES
a39bcbfac0e0        tomcat1             "catalina.sh run"   5 seconds ago
Up 4 seconds        0.0.0.0:11055->8080/tcp   vigilant_swanson
```

You can watch the result on firefox browser.



---

5. Build a tomcat Image:

Create a Docker file as below. Use an available Debian Image and add JDK and Tomcat zip files from our Downloads.

```
osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ cat Dockerfile
FROM debian:stretch
WORKDIR /opt/
ADD jdk-8u162-linux-x64.tar.gz /opt/
ADD apache-tomcat-8.5.27.tar.gz /opt/

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker image build -t tomcat2
.
Sending build context to Docker daemon  199.4MB
Step 1/4 : FROM debian:stretch
stretch: Pulling from library/debian
Digest:
sha256:c908a4fcb2b2a1953bd40ebc12d9a4116868d72540efc27502ee6c2395b8a1e9
Status: Downloaded newer image for debian:stretch
 ---> 2b98c9851a37
Step 2/4 : WORKDIR /opt/
 ---> Using cache
 ---> ab57e41dcde0
Step 3/4 : ADD jdk-8u162-linux-x64.tar.gz /opt/
 ---> Using cache
 ---> 824f9be4d745
Step 4/4 : ADD apache-tomcat-8.5.27.tar.gz /opt/
```

```
 ---> Using cache
 ---> 194ce98bd1fb
Successfully built 194ce98bd1fb
Successfully tagged tomcat2:latest

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
debian              stretch             2b98c9851a37        4 weeks ago
100MB

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker image build -t tomcat2
.
Sending build context to Docker daemon  199.4MB
Step 1/4 : FROM debian:stretch
stretch: Pulling from library/debian
Digest:
sha256:c908a4fcb2b2a1953bd40ebc12d9a4116868d72540efc27502ee6c2395b8a1e9
Status: Downloaded newer image for debian:stretch
 ---> 2b98c9851a37
Step 2/4 : WORKDIR /opt/
 ---> Using cache
 ---> ab57e41dcde0
Step 3/4 : ADD jdk-8u162-linux-x64.tar.gz /opt/
 ---> Using cache
 ---> 824f9be4d745
Step 4/4 : ADD apache-tomcat-8.5.27.tar.gz /opt/
 ---> Using cache
 ---> 194ce98bd1fb
Successfully built 194ce98bd1fb
Successfully tagged tomcat2:latest


osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker container run -it
tomcat2
root@e4b8cadb8842:/opt# ls
apache-tomcat-8.5.27  jdk1.8.0_162
root@e4b8cadb8842:/opt# ls apache-tomcat-8.5.27/
LICENSE  RELEASE-NOTES bin   lib   temp     work
NOTICE       RUNNING.txt     conf  logs  webapps
root@e4b8cadb8842:/opt# ls jdk1.8.0_162/
COPYRIGHT    THIRDPARTYLICENSEREADME-JAVAFX.txt  db       jre  release
LICENSE      THIRDPARTYLICENSEREADME.txt      include     lib  src.zip
README.html  bin                    javafx-src.zip  man
root@e4b8cadb8842:/opt# exit
exit

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ cat Dockerfile
FROM debian:stretch
ADD jdk-8u162-linux-x64.tar.gz /opt/
ADD apache-tomcat-8.5.27.tar.gz /opt/
ENV JAVA_HOME=/opt/jdk1.8.0_162
ENV PATH $JAVA_HOME/bin:$PATH
EXPOSE 8080
```

```
CMD ["/opt/apache-tomcat-8.5.27/bin/catalina.sh" , "run"]

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker image build -t tomcat3
.
<<output not shown>>

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker container run -d -p
11077:8080 tomcat3
<<output not shown>>
```

---

6. Sync a folder available in tomcat folder with webapps to deploy the application.

```
osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ cat Dockerfile
FROM debian:stretch
WORKDIR /opt/
ADD jdk-8u162-linux-x64.tar.gz /opt/
ADD apache-tomcat-8.5.27.tar.gz /opt/
ENV JAVA_HOME=/opt/jdk1.8.0_162
ENV PATH $JAVA_HOME/bin:$PATH
ADD ./web/*.war /opt/apache-tomcat-8.5.27/webapps/
EXPOSE 8080
CMD ["/opt/apache-tomcat-8.5.27/bin/catalina.sh" , "run"]

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker image build -t tomcat4
.
Sending build context to Docker daemon  199.4MB
Step 1/9 : FROM debian:stretch
 ---> 2b98c9851a37
Step 2/9 : WORKDIR /opt/
 ---> Using cache
 ---> ab57e41dcde0
Step 3/9 : ADD jdk-8u162-linux-x64.tar.gz /opt/
 ---> Using cache
 ---> 824f9be4d745
Step 4/9 : ADD apache-tomcat-8.5.27.tar.gz /opt/
 ---> Using cache
 ---> 194ce98bd1fb
Step 5/9 : ENV JAVA_HOME=/opt/jdk1.8.0_162
 ---> Using cache
 ---> a058d5299e88
Step 6/9 : ENV PATH $JAVA_HOME/bin:$PATH
 ---> Using cache
 ---> 5a9e1b24b0bd
Step 7/9 : ADD ./web/*.war /opt/apache-tomcat-8.5.27/webapps/
 ---> 0c051a7d5d2d
Step 8/9 : EXPOSE 8080
 ---> Running in 4687974faae6
```

```
Removing intermediate container 4687974faae6
 ---> 36d90e40015a
Step 9/9 : CMD ["/opt/apache-tomcat-8.5.27/bin/catalina.sh" , "run"]
 ---> Running in 43b3d699bbfd
Removing intermediate container 43b3d699bbfd
 ---> 275c390235cb
Successfully built 275c390235cb
Successfully tagged tomcat4:latest

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker container run -d -p
11077:8080 tomcat4
caf30bbc90e34d6453e1ea5990e9248875bfa065edd984441fbaf72eafa3fa2f

osgdev@TG-DevOps-OS004:~/dockerlab/tomcat$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                    NAMES
caf30bbc90e3        tomcat4             "/opt/apache-tomcat-…"   5
seconds ago       Up 4 seconds        0.0.0.0:11077->8080/tcp
flamboyant_sinoussi
```

---

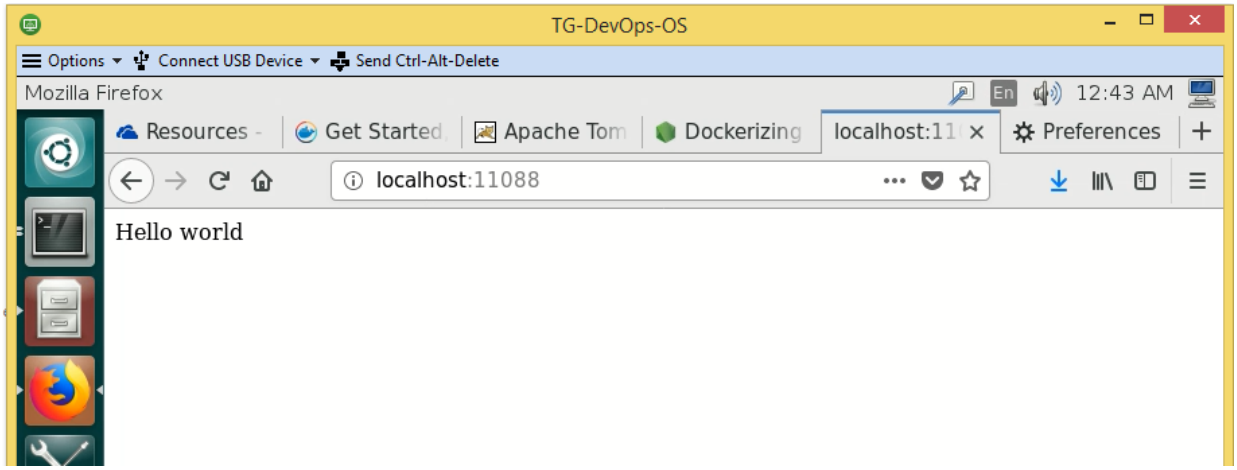
7. Dockerizing a Node.js webapp

```
osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ cat package.json
{
  "name": "docker_web_app",
  "version": "1.0.0",
  "description": "Node.js on Docker",
  "author": "First Last <first.last@example.com>",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.16.1"
```

```
  }
}

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ cat server.js
'use strict';

const express = require('express');

// Constants
const PORT = 8080;
const HOST = '0.0.0.0';

// App
const app = express();
app.get('/', (req, res) => {
  res.send('Hello world\n');
});

app.listen(PORT, HOST);
console.log(`Running on http://${HOST}:${PORT}`);

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ cat .dockerignore
node_modules
npm-debug.log

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ cat Dockerfile
FROM node:carbon

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json
are copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm install --only=production

# Bundle app source
COPY . .

EXPOSE 8080
CMD [ "npm", "start" ]

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ ls -a
.  ..  Dockerfile  .dockerignore  package.json  server.js

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ docker image pull node:carbon
carbon: Pulling from library/node
f2b6b4884fc8: Pull complete
4fb899b4df21: Pull complete
```

```
74eaa8be7221: Pull complete
2d6e98fe4040: Pull complete
452c06dec5fa: Pull complete
7b3c215894de: Pull complete
094529398b79: Pull complete
449fe646e95b: Pull complete
Digest:
sha256:26e4c77f9f797c3993780943239fa79419f011dd93ae4e0097089e2145aeaa24
Status: Downloaded newer image for node:carbon

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ docker build -t nodewebapp .
Sending build context to Docker daemon   5.12kB
Step 1/7 : FROM node:carbon
 ---> 4635bc7d130c
Step 2/7 : WORKDIR /usr/src/app
Removing intermediate container fa8a8e22da58
 ---> 54093eaa7939
Step 3/7 : COPY package*.json ./
 ---> 26c01006c219
Step 4/7 : RUN npm install
 ---> Running in 8f3cad565ab7
npm notice created a lockfile as package-lock.json. You should commit
this file.
npm WARN docker_web_app@1.0.0 No repository field.
npm WARN docker_web_app@1.0.0 No license field.

added 50 packages in 2.706s
Removing intermediate container 8f3cad565ab7
 ---> 3832653a88f8
Step 5/7 : COPY . .
 ---> 7544071f7d30
Step 6/7 : EXPOSE 8080
 ---> Running in 73abff25c7a3
Removing intermediate container 73abff25c7a3
 ---> d2ae97756053
Step 7/7 : CMD [ "npm", "start" ]
 ---> Running in b5b184c0c822
Removing intermediate container b5b184c0c822
 ---> 56d0cef7510c
Successfully built 56d0cef7510c
Successfully tagged nodewebapp:latest

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
nodewebapp          latest              56d0cef7510c        24 seconds
ago      675MB
node                carbon              4635bc7d130c        11 days ago
673MB

osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ docker container run -p
11088:8080 -d nodewebapp
5f4b4c74e7890fb7f41b8eeae13b15a04e023bdec4e25b5384d58242f02c53cb
```

```
osgdev@TG-DevOps-OS004:~/dockerlab/nodejs$ docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                NAMES
5f4b4c74e789        nodewebapp          "npm start"              4
seconds ago       Up 3 seconds        0.0.0.0:11088->8080/tcp
frosty_aryabhata
```



---

8. Let us launch the applications inside these images as services. Applications are available in the form of following images:

```
osgdev@TG-DevOps-OS004:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
nodewebapp          latest              56d0cef7510c        4 days ago
675MB
tomcat4             latest              275c390235cb        4 days ago
498MB
hellojava6          latest              0d9a1c04714b        4 days ago
643MB
sayhello            latest              02b691805eb2        4 days ago
150MB
whale001            latest              706bfd241e32        4 days ago
247MB
```

---

9. Services are launched in Docker Swarm. Hence shall initialize a Docker Swarm.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker swarm init
Swarm initialized: current node (ge1tu2ymefv7a8i1o8yf9mryg) is now a
manager.
```

To add a worker to this swarm, run the following command:

```
    docker swarm join --token SWMTKN-1-
3e5q4co5metaymj0bvjplmt99d9kiqyf7vfwpsfk9mhk13y804-
0a4y5c3q8r8clccwyd7xdvv6m 10.199.0.104:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

Note: Using the command provided in the command response worker nodes (installed with Docker) can be added as worker node to Docker Swarm resulting in a server cluster to launch the application.

Check the new networks created for this.

Networks before initiating docker swarm, was seen during docker network part.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9af5ffc53ff5        bridge              bridge              local
2f17cc107ea7        host                host                local
23c983327ebe        none                null                local
```

After the creation of Docker Swarm using "docker swarm init" command:

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9af5ffc53ff5        bridge              bridge              local
bdc5b03153da        docker_gwbridge     bridge              local
2f17cc107ea7        host                host                local
xwupecx1xmzm        ingress             overlay             swarm
23c983327ebe        none                null                local
```

---

10. Compose an YAML file which would launch "hellojava" image as a service.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ cat docker-compose.yaml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: hellojava6
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "80:80"
    networks:
```

```
      - webnet
networks:
  webnet:
```

Note: Above YAML file uses "hellojava6" image for deployment on servers. Create 5 replica of services. Uses 10% of CPU and 50MB of memory. If any container (holding the service) creation fails it will be restarted. All container ports (80) are mapped to same port (80) on system, and all containers are connected to "webnet" network.

---

11. Start docker stack which will create

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker stack deploy -c docker-
compose.yaml hello-java
Creating network hello-java_webnet
Creating service hello-java_web
```

Note: Create a overlay network for connectivity between containers "hello-java_webnet"

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9af5ffc53ff5        bridge              bridge              local
bdc5b03153da        docker_gwbridge     bridge              local
vclzg2htpu4w        hello-java_webnet   overlay             swarm
2f17cc107ea7        host                host                local
xwupecx1xmzm        ingress             overlay             swarm
23c983327ebe        none                null                local
```

Note: Created a service "hello-java_web"

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS
a7btp3xkhhpt        hello-java_web      replicated          0/5
hellojava6:latest   *:80->80/tcp
```

List of tasks corresponding to service "hello-java_web". Observe that 5 service replicas are created. All are exited as they have short life.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ps hello-java_web
ID                  NAME                IMAGE               NODE
DESIRED STATE       CURRENT STATE             ERROR               PORTS
f9abdle2kvq6        hello-java_web.1    hellojava6:latest   TG-DevOps-
OS004     Shutdown            Complete 43 seconds ago
m6jd4kgtun0b        hello-java_web.2    hellojava6:latest   TG-DevOps-
OS004     Shutdown            Complete 41 seconds ago
k49rn7fvuxjd        hello-java_web.3    hellojava6:latest   TG-DevOps-
OS004     Shutdown            Complete 33 seconds ago
rxitkvyl3jra        hello-java_web.4    hellojava6:latest   TG-DevOps-
OS004     Shutdown            Complete 33 seconds ago
```

```
098r5yfslvml        hello-java_web.5    hellojava6:latest   TG-DevOps-
OS004     Shutdown            Complete 37 seconds ago
```

These tasks are actually held in containers. None of them are running and all are exited.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS                      PORTS               NAMES
529cc29cd419        hellojava6:latest   "/bin/sh -c 'java -j…"   5
minutes ago       Exited (0) 5 minutes ago                     hello-
java_web.4.rxitkvyl3jras5habzmw7hb58
67328a709254        hellojava6:latest   "/bin/sh -c 'java -j…"   5
minutes ago       Exited (0) 5 minutes ago                     hello-
java_web.3.k49rn7fvuxjdqpq9big1h10k0
8e665dfc7236        hellojava6:latest   "/bin/sh -c 'java -j…"   5
minutes ago       Exited (0) 5 minutes ago                     hello-
java_web.5.098r5yfslvmlk5btlk5h27yrz
0703aa3a7ba3        hellojava6:latest   "/bin/sh -c 'java -j…"   5
minutes ago       Exited (0) 5 minutes ago                     hello-
java_web.2.m6jd4kgtun0bwn620zx9kki9f
d475d10f9f52        hellojava6:latest   "/bin/sh -c 'java -j…"   6
minutes ago       Exited (0) 5 minutes ago                     hello-
java_web.1.f9abdle2kvq6nyz6l5i3qjtao
```

Remove the service:

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker service rm hello-java_web
hello-java_web

osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS
```

Remove the stopped containers

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container prune
```

Remove the overlay network created

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker network rm hello-java_webnet
hello-java_webnet
```

---

12. Create similar service stack with "sayhello" image. Create a docker-compose YAML file.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ cat docker-compose-sayhello.yaml
version: "3"
```

```
services:
  web:
    # replace username/repo:tag with your name and image details
    image: sayhello
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "11055:80"
    networks:
      - webnet
networks:
  webnet:

osgdev@TG-DevOps-OS004:~/dockerlab$ docker stack deploy -c docker-
compose-sayhello.yaml sayhello
Creating network sayhello_webnet
Creating service sayhello_web

osgdev@TG-DevOps-OS004:~/dockerlab$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9af5ffc53ff5        bridge              bridge              local
bdc5b03153da        docker_gwbridge     bridge              local
2f17cc107ea7        host                host                local
xwupecx1xmzm        ingress             overlay             swarm
23c983327ebe        none                null                local
39dqwg1v9rv1        sayhello_webnet     overlay             swarm

osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS
ov0pu16c4xon        sayhello_web        replicated          5/5
sayhello:latest     *:11055->80/tcp

osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ps sayhello_web
ID                  NAME                IMAGE               NODE
DESIRED STATE       CURRENT STATE               ERROR
PORTS
yjdoa4afr5t5        sayhello_web.1      sayhello:latest     TG-DevOps-
OS004     Running             Running about a minute ago
uyihck8x7zip        sayhello_web.2      sayhello:latest     TG-DevOps-
OS004     Running             Running about a minute ago
liu7hqt9t9vy        sayhello_web.3      sayhello:latest     TG-DevOps-
OS004     Running             Running about a minute ago
xiau0i0lrkrv        sayhello_web.4      sayhello:latest     TG-DevOps-
OS004     Running             Running about a minute ago
0z80sbkb1fwq        sayhello_web.5      sayhello:latest     TG-DevOps-
OS004     Running             Running about a minute ago
```

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
f50883c93ad7        sayhello:latest     "python app.py"     About a
minute ago   Up About a minute   80/tcp
sayhello_web.3.liu7hqt9t9vyzhlpa1jbwmgkt
1655c20aa3df        sayhello:latest     "python app.py"     About a
minute ago   Up About a minute   80/tcp
sayhello_web.5.0z80sbkb1fwqredo23ezc3vsx
b90d6b635561        sayhello:latest     "python app.py"     About a
minute ago   Up About a minute   80/tcp
sayhello_web.2.uyihck8x7ziptizkglovdxzy5
c96539e380c8        sayhello:latest     "python app.py"     About a
minute ago   Up About a minute   80/tcp
sayhello_web.4.xiau0i0lrkrvtnoe9nct73xjz
7b52524fb15f        sayhello:latest     "python app.py"     About a
minute ago   Up About a minute   80/tcp
sayhello_web.1.yjdoa4afr5t5076v4hk6l7vy4
```

---

13. Removing the service

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS
n8x5kvzjdw0k        sayhello_web        replicated          5/5
sayhello:latest     *:11055->80/tcp
```

Note: Any listed service can be removed by using "service rm" command

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker service rm sayhello_web
sayhello_web
```

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS
```

However this would have only removed the service, but the "webnet" network continue to exist.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9af5ffc53ff5        bridge              bridge              local
bdc5b03153da        docker_gwbridge     bridge              local
2f17cc107ea7        host                host                local
xwupecx1xmzm        ingress             overlay             swarm
23c983327ebe        none                null                local
z19kekziqwz5        sayhello_webnet     overlay             swarm
```

All corresponding containers would exit as the service is stopped.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

osgdev@TG-DevOps-OS004:~/dockerlab$ docker container prune
```

---

14. Removing Stack of Services

Create the service again. Note that removing the service has only removed the service retaining the network created to connect the stack of services. Hence sayhello_webnet is not created again.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker stack deploy -c docker-
compose-sayhello.yaml sayhello
Creating service sayhello_web
```

But when you remove the stack, service along with network is removed.

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker stack rm sayhello
Removing service sayhello_web
Removing network sayhello_webnet

osgdev@TG-DevOps-OS004:~/dockerlab$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS

osgdev@TG-DevOps-OS004:~/dockerlab$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9af5ffc53ff5        bridge              bridge              local
bdc5b03153da        docker_gwbridge     bridge              local
2f17cc107ea7        host                host                local
xwupecx1xmzm        ingress             overlay             swarm
23c983327ebe        none                null                local
```

---

15. Finally you can also down the swarm. Since bringing down the manager would end the swarm you need the flag --force

```
osgdev@TG-DevOps-OS004:~/dockerlab$ docker swarm leave
Error response from daemon: You are attempting to leave the swarm on a
node that is participating as a manager. Removing the last manager erases
all current state of the swarm. Use `--force` to ignore this message.

osgdev@TG-DevOps-OS004:~/dockerlab$ docker swarm leave --force
Node left the swarm.
```

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.
