**First install Docker on Local machine. We chose Ubuntu 14.04.5.**

1. **Update the apt package index:**

$ sudo apt-get update

**2. Install packages to allow apt to use a repository over HTTPS:**

$ sudo apt-get install \  
 apt-transport-https \  
 ca-certificates \  
 curl \  
 Software-properties-common

**3. Add Docker’s official GPG key:**

3. $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

**4. Verify key with fingerprint:**

$ sudo apt-key fingerprint 0EBFCD88

**Should return:**

pub 4096R/0EBFCD88 2017-02-22  
 Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88  
uid Docker Release (CE deb) <docker@docker.com>  
sub 4096R/F273FCD8 2017-02-22

**5. Run command to setup stable repo:**

**$ sudo add-apt-repository \  
 "deb [arch=amd64] https://download.docker.com/linux/ubuntu \  
 $(lsb\_release -cs) \  
 stable"**

**6. Actual install:**

**$ sudo apt-get update**

**7. Install the *latest version* of Docker CE**

$ sudo apt-get install docker-ce

**8. Verify that Docker CE is installed correctly by running the hello-world image:**

$ sudo docker run hello-world

**You should see:** Hello from Docker! And a description of the steps the program ran to generate the message.

**9. Install PIP:**

$ sudo apt-get install python-pip

**Verify pip install with:**

pip --version

**10. Create an empty directory (I called it Docker\_dir):**

MKDIR <directory name>

**CD into that dir**

**11. Create 3 files inside this directory: Dockerfile, app.py, requirements.txt**

**11a. Create Dockerfile:**

touch Dockerfile app.py requirements.txt

**11b. Add necessary permissions:**

Chmod 0755 Dockerfile app.py requirements.txt

**11c. Dockerfile contents:**

# 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  
COPY . /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"]

**11d.requirements.txt contents:**

Flask  
Redis

**11e. App.py contents:**

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)

**12. Run command to install redis and flask libraries:**

pip install -r requirements.txt

**13. Build the app:ls command should show the three files we created. Then run**

Sudo docker build -t friendlyhello .

**An image will build.Run the below command to display the newly built image.**

Sudo docker image ls

**14. Run the app:**

**sudo docker run -p 4000:80 friendlyhello**

**You should see a message that Python is serving your app at http://0.0.0.0:80. But that message is coming from inside the container, which doesn’t know you mapped port 80 of that container to 4000, making the correct URL http://localhost:4000.**

**Go to that URL in a web browser to see the display content served up on a web page:**

Hello world!

Hostname: 73fde6e646bf

Visits: Cannot connect to Redis, counter disabled

**You can also use the curl command in a shell to view the same content:**

$ curl http://localhost:4000  
  
<h3>Hello World!</h3><b>Hostname:</b> 8fc990912a14<br/><b>Visits:</b> <i>cannot connect to Redis, counter disabled</i>

## Share your image

1. Log in with docker ID:

$ sudo docker login

### Tag the image

docker tag image username/repository:tag

My test example is:

docker tag image ae637/testrepo:test

### Publish the image

docker push username/repository:tag

**My example:**

docker push ae637/testrepo:test

**Once complete, the results of this upload are publicly available. If you log in to** [**Docker Hub**](https://hub.docker.com/)**, you see the new image there, with its pull command.**

### Pull and run the image from the remote repository

From now on, you can use docker run and run your app on any machine with this command:

docker run -p 4000:80 username/repository:tag

**My Example:**

docker run -p 4000:80 ae637/testrepo:test

No matter where docker run executes, it pulls your image, along with Python and all the dependencies from requirements.txt, and runs your code. It all travels together in a neat little package, and you don’t need to install anything on the host machine for Docker to run it.