Name	Pratiksha Naik
Class	BE IT A
Roll Number	29

Devops Assignments

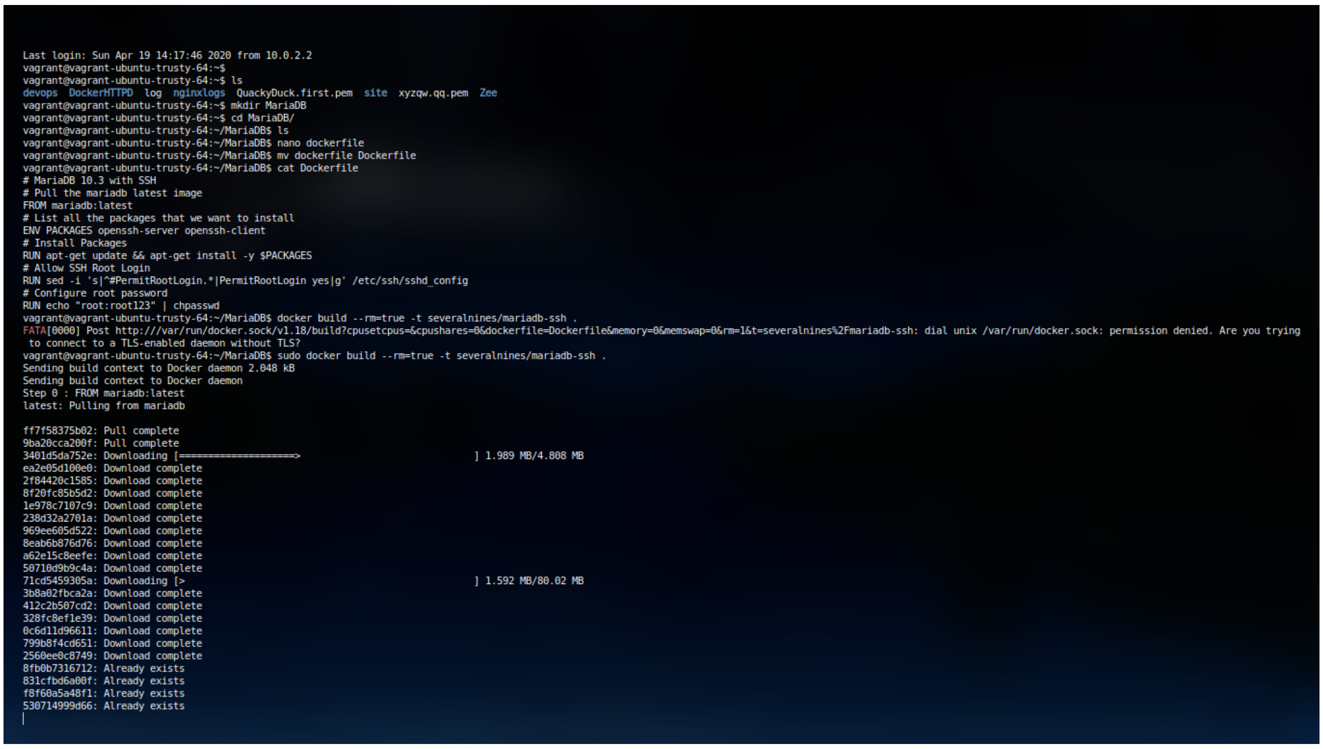
Assignment 2

1. Dockerfile to host a DB server.
2. Dockerfile to run a python3 program.

Dockerfile to host a DB server.

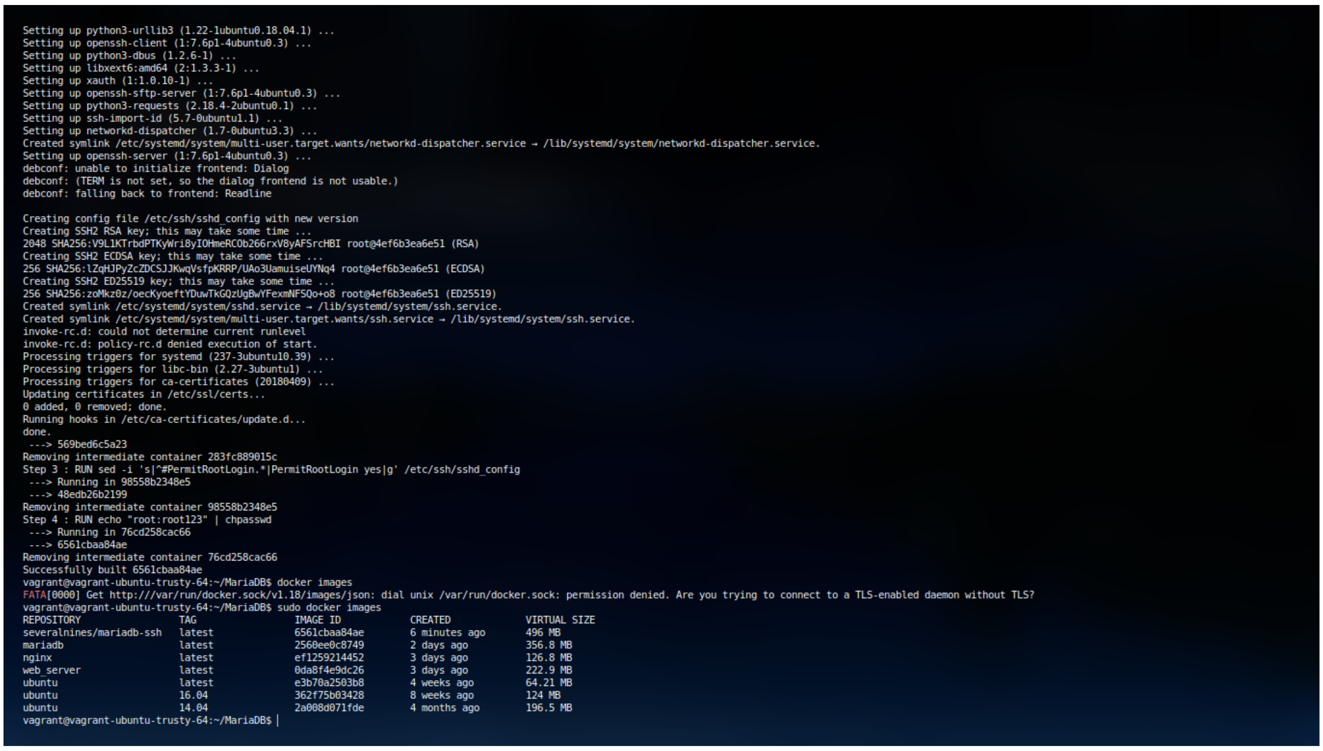
`Create a Dockerfile to basically pull MARIADB and required fs from server`

```
# MariaDB 10.3 with SSH
# Pull the mariadb latest image
FROM mariadb:latest
# List all the packages that we want to install
ENV PACKAGES openssh-server openssh-client
# Install Packages
RUN apt-get update && apt-get install -y $PACKAGES
# Allow SSH Root Login
RUN sed -i 's|^#PermitRootLogin.*|PermitRootLogin yes|g'
/etc/ssh/sshd_config
# Configure root password
RUN echo "root:root123" | chpasswd
```



```
Last login: Sun Apr 19 14:17:46 2020 from 10.0.2.2
vagrant@vagrant-ubuntu-trusty-64:~$
vagrant@vagrant-ubuntu-trusty-64:~$ ls
devops  DockerHTTPD  log  nginxlogs  QuackyDuck.first.pem  site  xyzqw.qq.pem  Zee
vagrant@vagrant-ubuntu-trusty-64:~$ mkdir MariaDB
vagrant@vagrant-ubuntu-trusty-64:~$ cd MariaDB/
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ ls
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ nano dockerfile
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ mv dockerfile Dockerfile
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ cat Dockerfile
# MariaDB 10.3 with SSH
# Pull the mariadb latest image
FROM mariadb:latest
# List all the packages that we want to install
ENV PACKAGES openssh-server openssh-client
# Install Packages
RUN apt-get update && apt-get install -y $PACKAGES
# Allow SSH Root Login
RUN sed -i 's|^#PermitRootLogin.*|PermitRootLogin yes|g' /etc/ssh/sshd_config
# Configure root password
RUN echo "root:root123" | chpasswd
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ docker build --rm=true -t severalnines/mariadb-ssh .
FATA[0000] Post http:///var/run/docker.sock/v1.18/build?cpusetcpus=&cpushares=0&dockerfile=Dockerfile&memory=0&memswap=0&rm=1&t=severalnines%2Fmariadb-ssh: dial unix /var/run/docker.sock: permission denied. Are you trying
 to connect to a TLS-enabled daemon without TLS?
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ sudo docker build --rm=true -t severalnines/mariadb-ssh .
Sending build context to Docker daemon 2.048 kB
Sending build context to Docker daemon
Step 0 : FROM mariadb:latest
latest: Pulling from mariadb

ff7f58375b02: Pull complete
9ba20cca200f: Pull complete
3401d5da752e: Downloading [===================>                              ] 1.989 MB/4.808 MB
ea2e05d100e0: Download complete
2f84420c1585: Download complete
8f20fc85b5d2: Download complete
1e978c7107c9: Download complete
238d32a2701a: Download complete
969ee605d522: Download complete
8eab6b876d76: Download complete
a62e15c8eefe: Download complete
50710d9b9c4a: Download complete
71cd5459305a: Downloading [>                                                 ] 1.592 MB/80.02 MB
3b8a02fbca2a: Download complete
412c2b507cd2: Download complete
328fc8ef1e39: Download complete
0c6d11d96611: Download complete
799b8f4cd651: Download complete
2560ee0c8749: Download complete
8fb0b7316712: Already exists
831cfbd6a00f: Already exists
f8f60a5a48f1: Already exists
530714999d66: Already exists
|
```

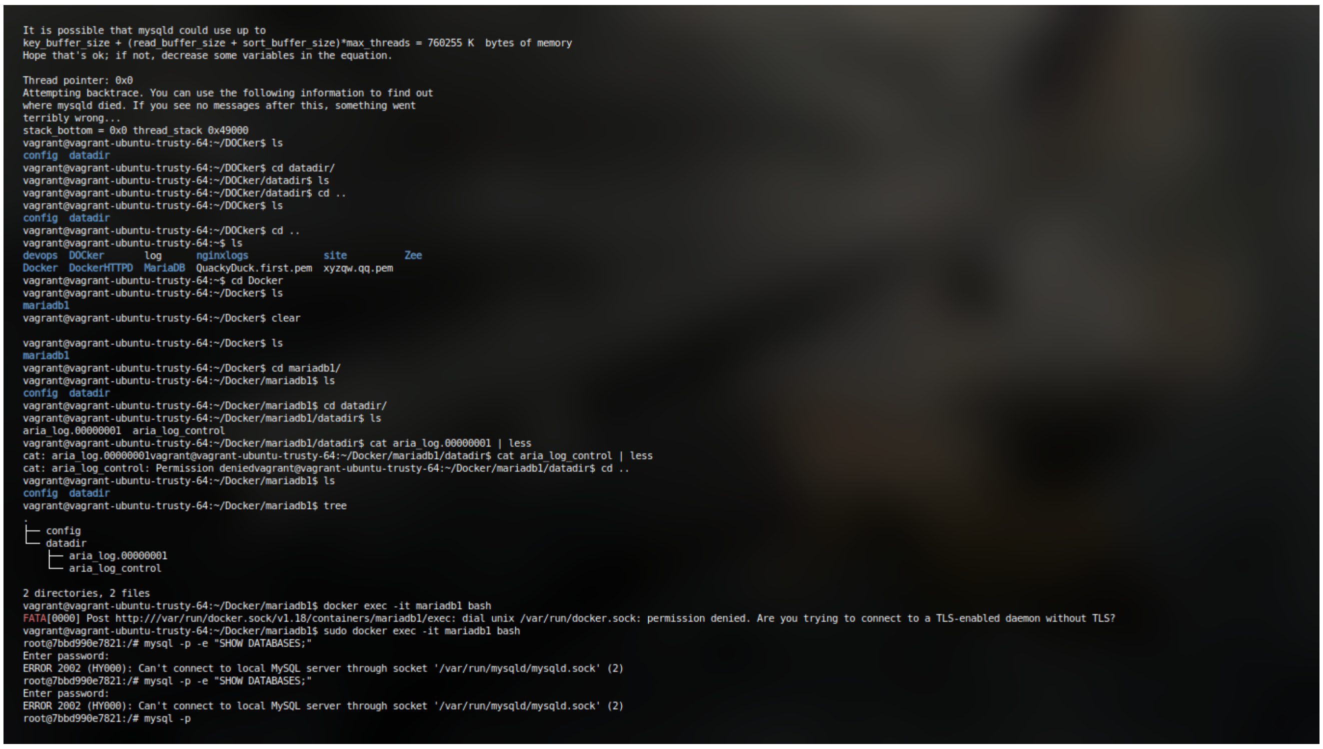
Then simply run the following to run container `docker build --rm=true -t severalnines/mariadb-ssh .` then `docker images`



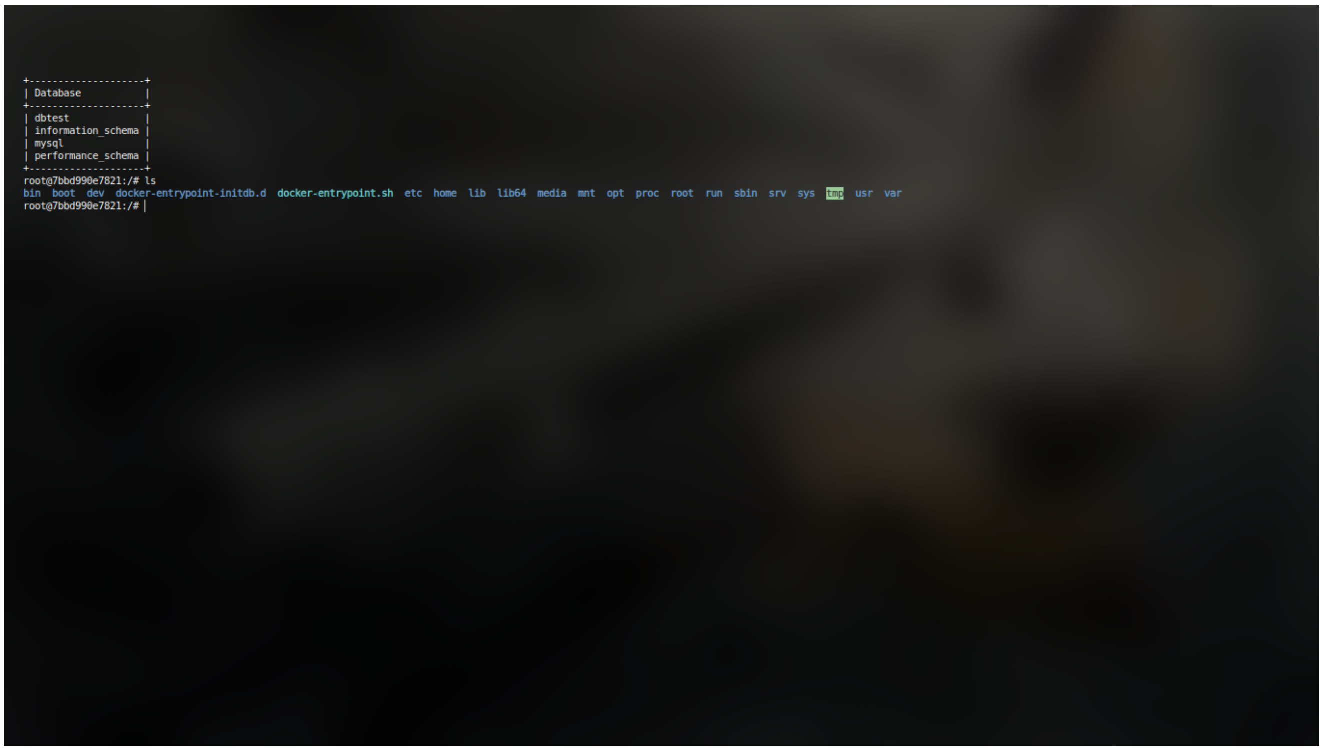
```
Setting up python3-urllib3 (1.22-1ubuntu0.18.04.1) ...
Setting up openssh-client (1:7.6p1-4ubuntu0.3) ...
Setting up python3-dbus (1.2.6-1) ...
Setting up libxext6:amd64 (2:1.3.3-1) ...
Setting up xauth (1:1.0.10-1) ...
Setting up openssh-sftp-server (1:7.6p1-4ubuntu0.3) ...
Setting up python3-requests (2.18.4-2ubuntu0.1) ...
Setting up ssh-import-id (5.7-0ubuntu1.1) ...
Setting up networkd-dispatcher (1.7-0ubuntu3.3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/networkd-dispatcher.service → /lib/systemd/system/networkd-dispatcher.service.
Setting up openssh-server (1:7.6p1-4ubuntu0.3) ...
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline

Creating config file /etc/ssh/sshd_config with new version
Creating SSH2 RSA key; this may take some time ...
2048 SHA256:V9L1KTrbdPTKyWri8yIOHmeRCOb266rxV8yAFSrcHBI root@4ef6b3ea6e51 (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:lZqHJPyZcZDCSJJKwqVsfpKRRP/UAo3UamuiseUYNq4 root@4ef6b3ea6e51 (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:zoMkz0z/oecKyoeftYDuwTkGQzUgBwYFexmNFSQo+o8 root@4ef6b3ea6e51 (ED25519)
Created symlink /etc/systemd/system/sshd.service → /lib/systemd/system/ssh.service.
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /lib/systemd/system/ssh.service.
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for systemd (237-3ubuntu10.39) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for ca-certificates (20180409) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
 ---> 569bed6c5a23
Removing intermediate container 283fc889015c
Step 3 : RUN sed -i 's|^#PermitRootLogin.*|PermitRootLogin yes|g' /etc/ssh/sshd_config
 ---> Running in 98558b2348e5
 ---> 48edb26b2199
Removing intermediate container 98558b2348e5
Step 4 : RUN echo "root:root123" | chpasswd
 ---> Running in 76cd258cac66
 ---> 6561cbaa84ae
Removing intermediate container 76cd258cac66
Successfully built 6561cbaa84ae
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ docker images
FATA[0000] Get http:///var/run/docker.sock/v1.18/images/json: dial unix /var/run/docker.sock: permission denied. Are you trying to connect to a TLS-enabled daemon without TLS?
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ sudo docker images
REPOSITORY                 TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
severalnines/mariadb-ssh   latest              6561cbaa84ae        6 minutes ago       496 MB
mariadb                    latest              2560ee0c8749        2 days ago          356.8 MB
nginx                      latest              ef1259214452        3 days ago          126.8 MB
web_server                 latest              0da8f4e9dc26        3 days ago          222.9 MB
ubuntu                     latest              e3b70a2503b8        4 weeks ago         64.21 MB
ubuntu                     16.04               362f75b03428        8 weeks ago         124 MB
ubuntu                     14.04               2a008d071fde        4 months ago        196.5 MB
vagrant@vagrant-ubuntu-trusty-64:~/MariaDB$ |
```

`cd ~/Docker mkdir datadir mkdir configure tail -1 /etc/mysql/my.cnf !includedir /etc/mysql/conf.d/` Then execute the following to run container.

```
docker run -d --name mariadb1 \
-p 33061:3306 \
-v ~/Docker/mariadb1/config:/etc/mysql/conf.d \
-v ~/Docker/mariadb1/datadir:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=root123 \
-e MYSQL_DATABASE=dbtest \
mariadb
```



```
It is possible that mysqld could use up to
key_buffer_size + (read_buffer_size + sort_buffer_size)*max_threads = 760255 K  bytes of memory
Hope that's ok; if not, decrease some variables in the equation.

Thread pointer: 0x0
Attempting backtrace. You can use the following information to find out
where mysqld died. If you see no messages after this, something went
terribly wrong...
stack_bottom = 0x0 thread_stack 0x49000
vagrant@vagrant-ubuntu-trusty-64:~/DOCker$ ls
config  datadir
vagrant@vagrant-ubuntu-trusty-64:~/DOCker$ cd datadir/
vagrant@vagrant-ubuntu-trusty-64:~/DOCker/datadir$ ls
vagrant@vagrant-ubuntu-trusty-64:~/DOCker/datadir$ cd ..
vagrant@vagrant-ubuntu-trusty-64:~/DOCker$ ls
config  datadir
vagrant@vagrant-ubuntu-trusty-64:~/DOCker$ cd ..
vagrant@vagrant-ubuntu-trusty-64:~$ ls
devops  DOCker       log      nginxlogs           site         Zee
Docker  DockerHTTPD  MariaDB  QuackyDuck.first.pem  xyzqw.qq.pem
vagrant@vagrant-ubuntu-trusty-64:~$ cd Docker
vagrant@vagrant-ubuntu-trusty-64:~/Docker$ ls
mariadb1
vagrant@vagrant-ubuntu-trusty-64:~/Docker$ clear

vagrant@vagrant-ubuntu-trusty-64:~/Docker$ ls
mariadb1
vagrant@vagrant-ubuntu-trusty-64:~/Docker$ cd mariadb1/
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1$ ls
config  datadir
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1$ cd datadir/
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1/datadir$ ls
aria_log.00000001  aria_log_control
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1/datadir$ cat aria_log.00000001 | less
cat: aria_log.00000001vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1/datadir$ cat aria_log_control | less
cat: aria_log_control: Permission deniedvagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1/datadir$ cd ..
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1$ ls
config  datadir
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1$ tree
.
├── config
└── datadir
    ├── aria_log.00000001
    └── aria_log_control

2 directories, 2 files
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1$ docker exec -it mariadb1 bash
FATA[0000] Post http:///var/run/docker.sock/v1.18/containers/mariadb1/exec: dial unix /var/run/docker.sock: permission denied. Are you trying to connect to a TLS-enabled daemon without TLS?
vagrant@vagrant-ubuntu-trusty-64:~/Docker/mariadb1$ sudo docker exec -it mariadb1 bash
root@7bbd990e7821:/# mysql -p -e "SHOW DATABASES;"
Enter password:
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2)
root@7bbd990e7821:/# mysql -p -e "SHOW DATABASES;"
Enter password:
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2)
root@7bbd990e7821:/# mysql -p
```



```
+--------------------+
| Database           |
+--------------------+
| dbtest             |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
root@7bbd990e7821:/# ls
bin  boot  dev  docker-entrypoint-initdb.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@7bbd990e7821:/# |
```

Question 2

Run a python program using DockerFile

Install flask (or requried dependencies for your programs) and write your respective code in file. Since you will be using pip for installing dependencies simply go ahead and write a requirements.txt as well

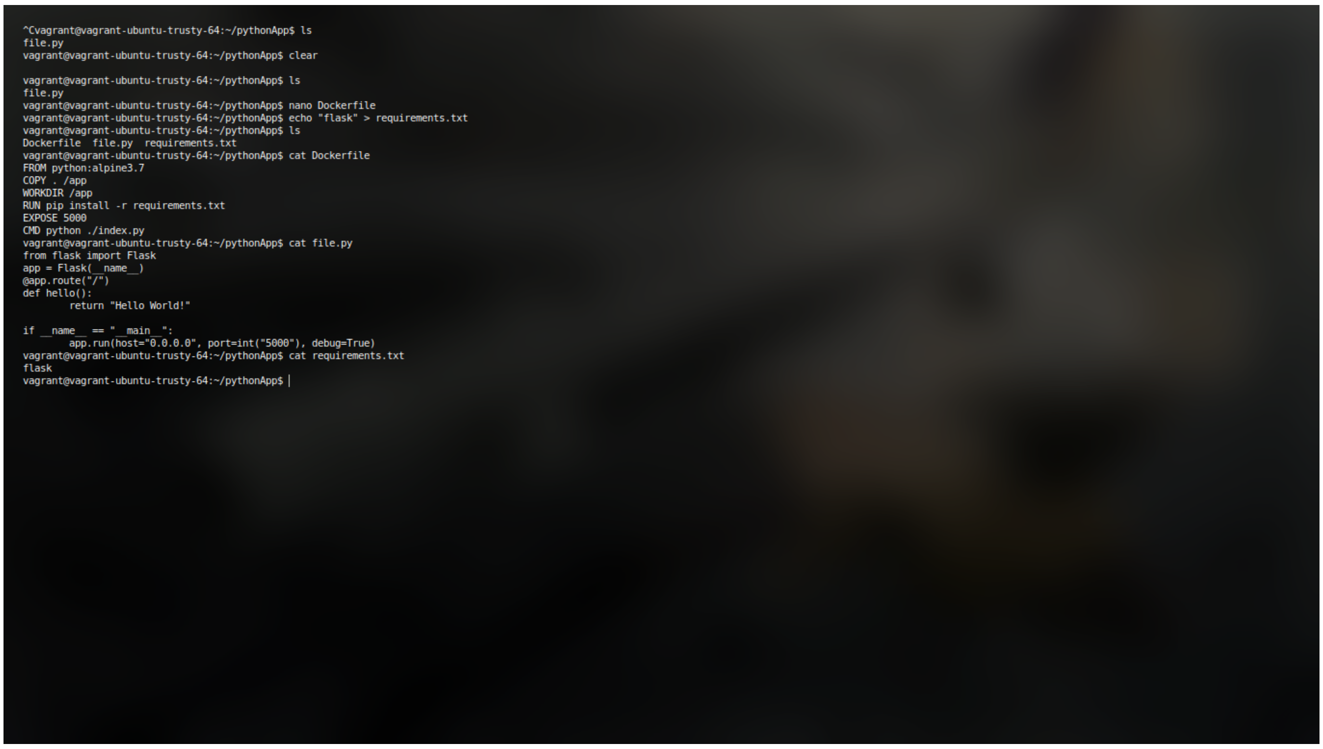
`echo "flask" > requirements.txt`

Code

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
    return "Hello World!"
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=int("5000"), debug=True)
```

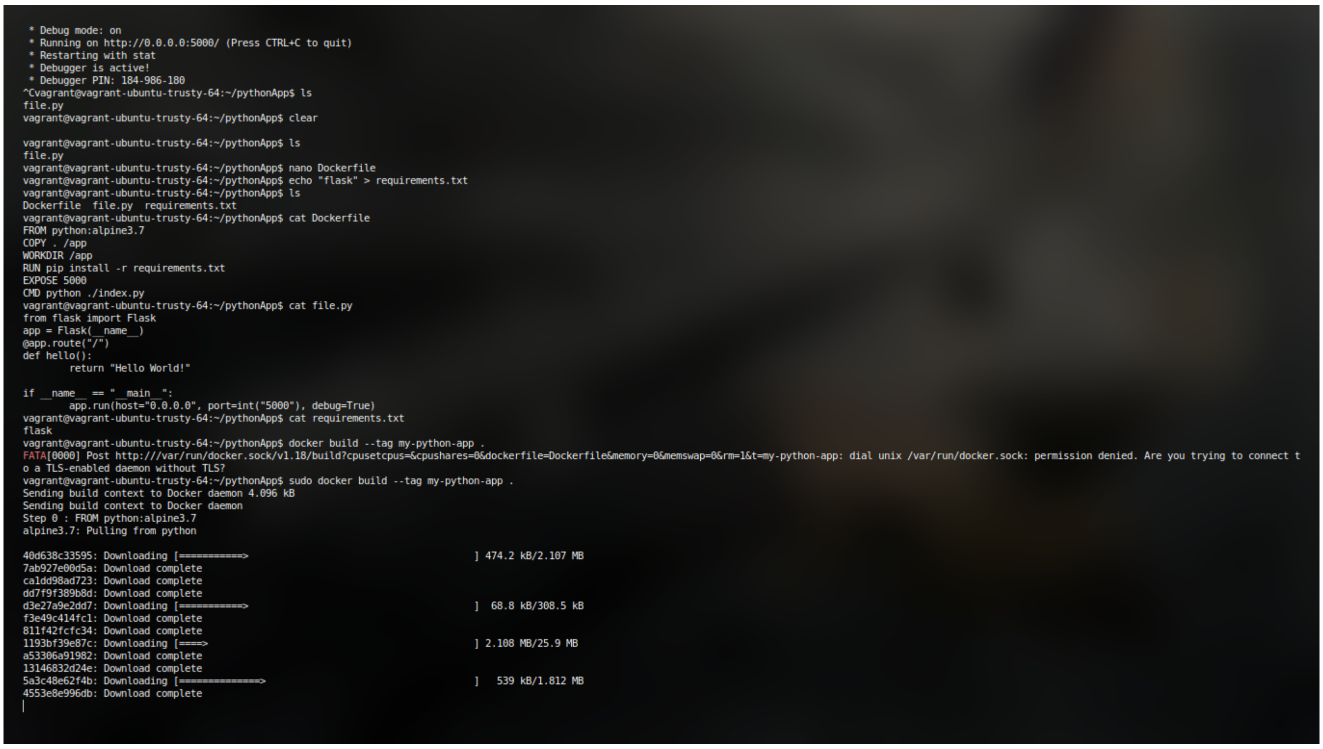
Now simply create a Dockerfile with the following content

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5000
CMD python3 ./file.py
```

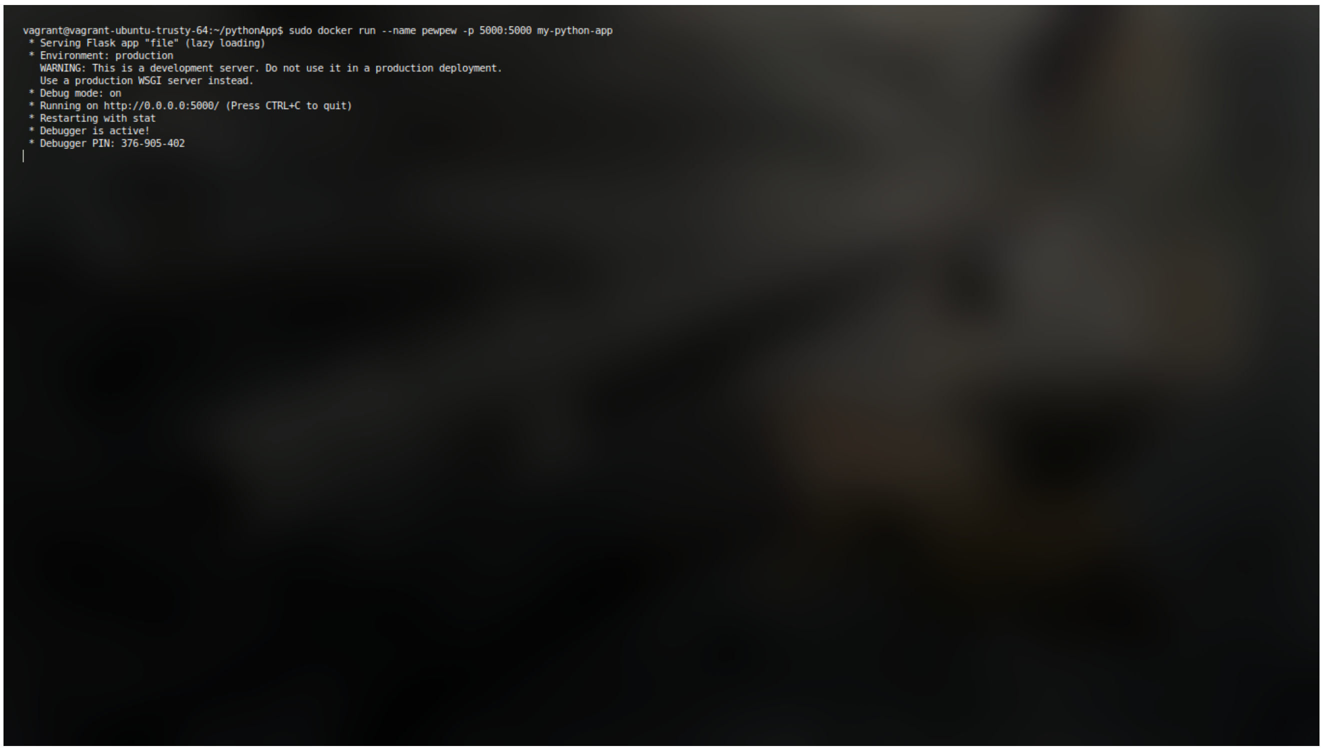


```
^Cvagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ ls
file.py
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ clear

vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ ls
file.py
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ nano Dockerfile
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ echo "flask" > requirements.txt
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ ls
Dockerfile  file.py  requirements.txt
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ cat Dockerfile
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5000
CMD python ./index.py
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ cat file.py
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
        return "Hello World!"

if __name__ == "__main__":
        app.run(host="0.0.0.0", port=int("5000"), debug=True)
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ cat requirements.txt
flask
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ |
```

Now all thats left is to build the image and run it.

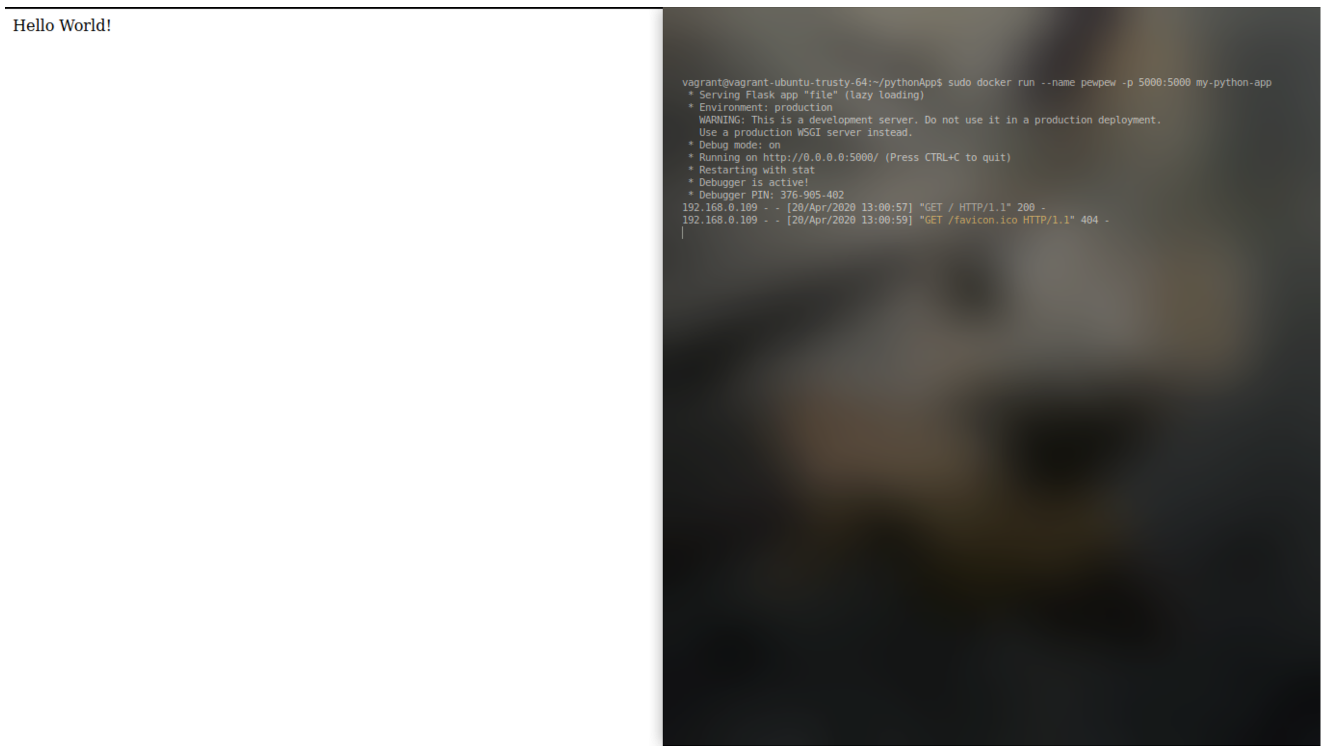


```
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 184-986-180
^Cvagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ ls
file.py
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ clear

vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ ls
file.py
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ nano Dockerfile
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ echo "flask" > requirements.txt
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ ls
Dockerfile  file.py  requirements.txt
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ cat Dockerfile
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5000
CMD python ./index.py
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ cat file.py
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
        return "Hello World!"

if __name__ == "__main__":
        app.run(host="0.0.0.0", port=int("5000"), debug=True)
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ cat requirements.txt
flask
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ docker build --tag my-python-app .
FATA[0000] Post http:///var/run/docker.sock/v1.18/build?cpusetcpus=&cpushares=0&dockerfile=Dockerfile&memory=0&memswap=0&rm=1&t=my-python-app: dial unix /var/run/docker.sock: permission denied. Are you trying to connect t
o a TLS-enabled daemon without TLS?
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ sudo docker build --tag my-python-app .
Sending build context to Docker daemon 4.096 kB
Sending build context to Docker daemon
Step 0 : FROM python:alpine3.7
alpine3.7: Pulling from python

40d638c33595: Downloading [===========>                                       ] 474.2 kB/2.107 MB
7ab927e00d5a: Download complete
ca1dd98ad723: Download complete
dd7f9f389b8d: Download complete
d3e27a9e2dd7: Downloading [===========>                                       ]  68.8 kB/308.5 kB
f3e49c414fc1: Download complete
811f42fcfc34: Download complete
1193bf39e87c: Downloading [====>                                              ] 2.108 MB/25.9 MB
a53306a91982: Download complete
13146832d24e: Download complete
5a3c48e62f4b: Downloading [==============>                                    ]   539 kB/1.812 MB
4553e8e996db: Download complete
```



```
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ sudo docker run --name pewpew -p 5000:5000 my-python-app
 * Serving Flask app "file" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 376-905-402
```

Docker container running like pewpew :D



Hello World!

```
vagrant@vagrant-ubuntu-trusty-64:~/pythonApp$ sudo docker run --name pewpew -p 5000:5000 my-python-app
 * Serving Flask app "file" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 376-905-402
192.168.0.109 - - [20/Apr/2020 13:00:57] "GET / HTTP/1.1" 200 -
192.168.0.109 - - [20/Apr/2020 13:00:59] "GET /favicon.ico HTTP/1.1" 404 -
```