**Containerization of sample Django application with gunicorn and ngnix**

Environment: Ubuntu-16.04 LTS

Step 1: Install the following

Pre-requisite :

1. Docker
2. Docker-compose

Document Link :

1. Docker
2. Docker-compose

Steps 2:

Creation of Docker file for Django application

|  |
| --- |
| FROM python:3.7  MAINTAINER Anirban Deb  COPY ./djangoModule /djangoModule  WORKDIR /djangoModule  RUN pip install -r requirements.txt && \  pip install gunicorn  COPY ./dockerfile/entrypoint.sh /entrypoint.sh  RUN chmod +x /entrypoint.sh  ENTRYPOINT ["/entrypoint.sh"] |

Create the entrypoint.sh script

|  |
| --- |
| #!/bin/bash  echo "Running command '$\*'"  exec /bin/bash -c "$\*" |

Step 3:

Create the ngnix docker file

|  |
| --- |
| FROM nginx:latest  RUN rm /etc/nginx/conf.d/default.conf  COPY ./dockerfile/ngnix/djangoproject.conf /etc/nginx/conf.d/ |

Create the ngnix configuration file

|  |
| --- |
| server{  listen 80;  location / {  proxy\_pass "http://{public\_ip}:8000";  }  } |

Step 3:

Creation of docker-compose file

|  |
| --- |
| #####  # Docker compose YAML file  #  # For documentation see: https://docs.docker.com/compose/yml/  #####  version: "3"  services:  python:  build:  context: .  dockerfile: dockerfile/Dockerfile  volumes:  - ./djangoModule:/djangoModule  ports:  - "8000:8000"  command: gunicorn -w 4 djangoModule.wsgi --bind 0.0.0.0:8000  ngnix:  build:  context: .  dockerfile: dockerfile/ngnix/Dockerfile  ports:  - "80:80" |

Step 4:

Copy the Django module in the current working directory and follow the below-working directory structure

|  |
| --- |
| .  ├── djangoModule  │ ├── db.sqlite3  │ ├── djangoModule  │ │ ├── asgi.py  │ │ ├── \_\_init\_\_.py  │ │ ├── \_\_pycache\_\_  │ │ │ ├── \_\_init\_\_.cpython-37.pyc  │ │ │ ├── settings.cpython-37.pyc  │ │ │ ├── urls.cpython-37.pyc  │ │ │ └── wsgi.cpython-37.pyc  │ │ ├── settings.py  │ │ ├── urls.py  │ │ └── wsgi.py  │ ├── manage.py  │ ├── myapp  │ │ ├── admin.py  │ │ ├── apps.py  │ │ ├── \_\_init\_\_.py  │ │ ├── migrations  │ │ │ └── \_\_init\_\_.py  │ │ ├── models.py  │ │ ├── \_\_pycache\_\_  │ │ │ ├── \_\_init\_\_.cpython-37.pyc  │ │ │ ├── urls.cpython-37.pyc  │ │ │ └── views.cpython-37.pyc  │ │ ├── tests.py  │ │ ├── urls.py  │ │ └── views.py  │ └── requirements.txt  ├── docker-compose.yml  └── dockerfile  ├── Dockerfile  ├── entrypoint.sh  └── ngnix  ├── djangoproject.conf  └── Dockerfile |

Check the requirements.txt

|  |
| --- |
| Django==3.0.4 |

Step 5: Go to the folder where the docker-compose.yml is present and run the below commands

|  |
| --- |
| [centos@docker-insstallation dockerApplication]$ docker-compose config  Output >> [NOTE : This checks for any systax issues in the docker-compose file]  services:  ngnix:  build:  context: /home/centos/dockerApplication  dockerfile: dockerfile/ngnix/Dockerfile  ports:  - 80:80/tcp  python:  build:  context: /home/centos/dockerApplication  dockerfile: dockerfile/Dockerfile  command: gunicorn -w 4 djangoModule.wsgi --bind 0.0.0.0:8000  ports:  - 8000:8000/tcp  volumes:  - /home/centos/dockerApplication/djangoModule:/djangoModule:rw  version: '3.0' |

Now once the systax is checked, run the docker-compose build command to create the docker images

|  |
| --- |
| [centos@docker-insstallation dockerApplication]$ sudo docker-compose build  Building python  Step 1/8 : FROM python:3.7  ---> 8e3336637d81  Step 2/8 : MAINTAINER Anirban Deb  ---> Using cache  ---> 230627d3a52f  Step 3/8 : COPY ./djangoModule /djangoModule  ---> Using cache  ---> 56cd4aa40fc0  Step 4/8 : WORKDIR /djangoModule  ---> Using cache  ---> 0db08c282936  Step 5/8 : RUN pip install -r requirements.txt && pip install gunicorn  ---> Using cache  ---> 28a68c9efd54  Step 6/8 : COPY ./dockerfile/entrypoint.sh /entrypoint.sh  ---> Using cache  ---> 133438de3025  Step 7/8 : RUN chmod +x /entrypoint.sh  ---> Using cache  ---> 3bc96fc064cd  Step 8/8 : ENTRYPOINT ["/entrypoint.sh"]  ---> Using cache  ---> e6dd2265ea79  Successfully built e6dd2265ea79  Successfully tagged dockerapplication\_python:latest  Building ngnix  Step 1/3 : FROM nginx:latest  ---> 6678c7c2e56c  Step 2/3 : RUN rm /etc/nginx/conf.d/default.conf  ---> Using cache  ---> eb35de0026dc  Step 3/3 : COPY ./dockerfile/ngnix/djangoproject.conf /etc/nginx/conf.d/  ---> Using cache  ---> f11796798d83  Successfully built f11796798d83  Successfully tagged dockerapplication\_ngnix:latest |

Now, run the docker-compose up to run the contaners

|  |
| --- |
| [centos@docker-insstallation dockerApplication]$ sudo docker-compose up  dockerapplication\_python\_1 is up-to-date  dockerapplication\_ngnix\_1 is up-to-date  Attaching to dockerapplication\_python\_1, dockerapplication\_ngnix\_1 |

Check the running of the container

|  |
| --- |
| [root@docker-insstallation centos]# docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  9567a52984a0 dockerapplication\_ngnix "nginx -g 'daemon of…" 17 hours ago Up 16 hours 0.0.0.0:80->80/tcp dockerapplication\_ngnix\_1  4d565896920d dockerapplication\_python "/entrypoint.sh guni…" 18 hours ago Up 16 hours 0.0.0.0:8000->8000/tcp dockerapplication\_python\_1 |

Step 6 : Check in the browser for the sample application

|  |
| --- |
|  |