

Part 1 Dockerfile&Containerize

```
systemctl stop packagekit
```

```
sudo yum -y update
```

```
systemctl stop httpd.service
```

```
mkdir lab
```

```
cd lab
```

```
cat > myapp.py
```

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
def test():
```

```
    return "<h1>Flask Server is working!</h1>"
```

```
if __name__ == "__main__":
```

```
    app.run(host="0.0.0.0",port=int("5000"), debug=True)
```

```
#Dockerfile olusturmadan once requirements.txt dosyasi olusturalim
```

```
cat > requirements.txt
```

```
flask
```

```
#simdi Dockerfile
```

```
cat > Dockerfile
```

```
FROM python:3.6
```

```
COPY ./*.py /
```

```
COPY ./requirements.txt /
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 8080
```

```
CMD ["python3.6","myapp.py"]
```

#python kodunu container imajina donustur

docker build -t myimage .

docker image ls

#kendi container'imizi yaratalim

docker container run -d -p 8080:5000 --name mycontainer myimage

localhost:8080

192.168.56.111.:8080

docker container stop mycontainer

docker container prune

#Sayisal loto icin python kodumuz

```
import random as rd
```

```
loto = []
```

```
while len(loto) < 6:
```

```
    sayi = rd.randint(1,99)
```

```
    if sayi not in loto:
```

```
        loto.append(sayi)
```

```
loto.sort()
```

```
print(loto)
```

#bu kodu flask kodumuzu entegre edelim #

bunun adi da myapp2.py olsun

mkdir lab2

cd lab2

```
cat >myapp2.py
```

```
from flask import Flask
```

```
import random as rd
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
def test():
```

```
    loto = []
```

```
    while len(loto) < 6:
```

```
        sayi = rd.randint(1,99)
```

```
        if sayi not in loto:
```

```
            loto.append(sayi)
```

```
    loto.sort()
```

```
    result=f'Winning Lottery numbers are: {loto}'
```

```
    return "<h1>Flask Server is working!</h1>" + result
```

```
if __name__ == "__main__":
```

```
    app.run(host="0.0.0.0",port=int("8000"), debug=True)
```

```
python3 myapp2.py
```

```
localhost:8000
```

```
#lab2 directory'sinde
```

```
cat > requirements.txt
```



flask

#Dockerfile

FROM python:3.6

COPY ./*.py /

COPY ./requirements.txt /

RUN pip install -r requirements.txt

EXPOSE 80

CMD ["python3.6", "myapp2.py"]

#python kodunu container imajina donustur

docker build -t myimage2 .

docker image ls

#kendi container'imizi yaratalim

docker container run -d -p 80:8000 --name mycontainer2 myimage2

localhost:80

192.168.56.111

docker image tag myimage2:latest eunvan/myimage2-may:1.1

docker image ls

docker login

docker image push eunvan/myimage2-may:1.1

#docker hub'a push ettigin imaji local repository'inden sil



```
docker image rm eunvan/myimage2-may:1.1
```

#simdi de docker hub'tan ayni imaji pull et

```
docker pull eunvan/myimage2-may:1.1
```

```
docker image ls
```

Part 2 Dockerfile&Containerize With Virtual Environment

```
sudo yum install python3
```

```
su -
```

```
yum -y install epel-release
```

```
yum install python-pip
```

```
pip3 install --upgrade pip
```

```
pip3 install pipenv
```

```
pipenv
```

```
exit
```

```
#root hesabından log out ol. admin ol
```

```
mkdir docker
```

```
cd docker
```

```
cat > myapp.py
```

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
def test():
```

```
    return "<h1>Server working!</h1>"
```

```
if __name__ == "__main__":
```

```
    app.run(host="0.0.0.0", port=int("8080"), debug=True)
```

```
#myapp.py dosyası docker klasörünün içinde olacak
```

```
#Virtual environment oluşturmamızın sebebi
```

```
#Python paketlerini virtual e environment üzerinde çalıştıracğız.
```

```
#Yüklediğimiz bütün paketleri bir klasöre yükleyecektir.
```

```
pipenv --three
```

```
 #(python 3.6.8 ile virtual bir environment oluşturduk bu komutla)
```

```
#Creating a Pipfile for this project
```

```
#pipfile oluşturuldu.
```

```
pipenv install flask
```

```
#check pipfile
```

```
pipenv shell
```

```
#(virtual environmenti aktif edelim) #docker klasorunun içindeyken

python3 myapp.py

#linux ta web browser'da localhost:8080 yazın

#Server working! yazısını göreceksiniz #peki bunu docker'dan nasıl çalıştırırız ?

#create Dockerfile:

cat > Dockerfile

# Use python:alpine3.6 for the base image

FROM python:3.6

# Copy the .py files in ./app from build's context to /app

COPY ./app/*.py /app/

# Copy the Pipfile in ./app from build's context to /app

COPY ./app/Pipfile /app

# Change the working directory to WORKDIR

WORKDIR /app

# Install pipenv

RUN pip install pipenv

# Install the environment

RUN pipenv install

# Expose port 8000

EXPOSE 8000

# Start the python application

CMD pipenv run python3.6 myapp.py

#Docker klasorunun içinde app diye bir klasör oluşturun.

mkdir app

#myapp.py ve pip file dosyasını bu app klasörünün içine koyun.

#Terminalde myapp.py uygulaması halen çalışıyor. Onu cırlt + c ile kapatın docker - -help

docker - -version

#dockerfile imajını docker klasöründe olacak

#docker klasoru icinde

docker build -t myapp .

#artık imajımız yaratıldı
```

docker image ls

#en üstte mypp'i göreceksin.

docker run -d -p 8000:8080 --name mycontainer myapp

#linux ta web browser'da localhost:8000 yazın

#Server working! yazısını göreceksiniz

#Şimdi lokalde değil, container'dan çalışıyor

docker image ls -a

docker container ls

#my app'ın mycontainer'in içinde çalıştığını göreceksiniz.

docker container inspect (container id)

#Container'in ne kadar özelliği varsa burada görebilirsiniz

docker search alpine

#docker hub'ta search edilir.

docker pull alpine

#docker hub'tan alpine'u kur.

#çalışan bir docker'ın ekranına nasıl bağlarız ?

docker exec -it (container id) /bin/sh

#Artık container'in içindeyiz whoami

ls

#exit ile çıkın

docker container stop (container id)

#sanal ortamdan cik

exit

#sanal ortami görüntüle

pipenv --venv

#sanal ortami sil

pipenv --rm

#ana klasore gel

cd /home/admin/

root ol

sudo su

rm -r -f /home/admin/docker/

Part 3 Dockerfile&Containerize An API

```
mkdir project
```

```
cd project
```

```
cat > calculator.py
```

```
from flask import Flask
```

```
from flask_restx import Resource, Api, reqparse
```

```
app=Flask(__name__)
```

```
api=Api(app)
```

```
carpim_parser=reqparse.RequestParser()
```

```
carpim_parser.add_argument('a',type=int,required=True,help='a sayisini yaz')
```

```
carpim_parser.add_argument('b',type=int,required=True,help='b sayisini yaz')
```

```
carpim_parser.add_argument('secim',choices=('toplama','cikarma','carpma','bolme'))
```

```
@api.route('/calculator')
```

```
class calculator(Resource):
```

```
    @api.expect(carpim_parser)
```

```
    def get(self):
```

```
        args=carpim_parser.parse_args()
```

```
        if args.secim=='toplama':
```

```
            return args.a+args.b
```

```
        elif args.secim=='cikarma':
```

```
            return args.a-args.b
```

```
        elif args.secim=='carpma':
```

```
            return args.a*args.b
```

```
        elif args.secim=='bolme' and args.b!=0:
```

```
            return args.a/args.b
```

```
        elif args.secim=='bolme' and args.b==0:
```

```
            return 'matematik bu islem olmaz'
```

```
        elif args.secim==None:
```

```
            return 'secim yap'
```

```
if __name__ == "__main__":
```

```
    app.run(host="0.0.0.0", port=int("8080"), debug=True)
```

```
cat > requirements.txt
```

```
flask
```

```
flask-restx
```

```
docker pull python:3.8
```

```
#Dockerfile
```

```
FROM python:3.8
```

```
COPY ./*.py /
```

```
COPY ./requirements.txt /
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 80
```

```
CMD ["python3.8","calculator.py"]
```

```
docker build -t calc .
```

```
docker run -d -p 80:8080 --name mycalculator calc
```

```
192.168.56.111
```