

Program 1: Build a Docker container from custom Docker file

Step 1: mkdir /DevOps/Program1 cd DevOps/Program1

Step 2: nano requirements.txt Flask

Step 3:nano app.py

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

Step 4: nano Dockerfile

```
# This is a Dockerfile
# It builds a container image for a simple Flask app
# Use a slim Python base image
FROM python:3.9-slim
# Set the working directory inside the container
WORKDIR /app
# Copy the requirements file and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
# Copy the application code
COPY ..
# Set Flask environment variables
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
ENV FLASK_RUN_PORT=5000
# Expose the port the Flask app will run on EXPOSE 5000
# Define the command to run the Flask application
CMD ["flask", "run"]
```

Step 5: docker build -t flask-app .

Step 6: docker run -p 5000:5000 flask-app

```
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~$ cd Dockerfile
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile$ cd prog1
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ ls
app.py  Dockerfile  requirements.txt
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ cat app.py
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, welcome to RVCE, MCA"

if __name__ == '__main__':
    app.run(host='localhost', port=5000)

1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ cat Dockerfile
# this is a Dockerfile
#use slim python base image
FROM python:3.9-slim

#set the working directory
WORKDIR /app

#upgrade pip and install system dependencies
RUN apt-get update && apt-get install -y build-essential && \
    pip install --upgrade pip setuptools

#copy the requirements file and install dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

#copy the application code
COPY .

#set Flask environment variables
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
ENV FLASK_RUN_PORT=5000

#expose the port
EXPOSE 5000

#command to run the Flask app
CMD ["flask", "run"]
```

```
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ cat ^[[200-requirements.txt
cat: ''$'\v33''[200-requirements.txt': No such file or directory
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ cat requirements.txt
flask
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ sudo docker build -t prog1 .
[sudo] password for 1RV24MC063_mahadevprasad_dl:
[+] Building 4.4s (11/11) FINISHED
  => [internal] load build definition from Dockerfile
  => transferring dockerfile: 657B
  => [internal] load metadata for docker.io/library/python:3.9-slim
  => [internal] load .dockerignore
  => transferring context: 28
  => [1/6] FROM docker.io/library/python:3.9-slim@sha256:545badabace9a958b98d3e272f0f0d46c0a1a389ac77e24c33f2e7b548ce1b6b
  => [internal] load build context
  => transferring context: 93B
  => CACHE [2/6] WORKDIR /app
  => CACHE [3/6] RUN apt-get update && apt-get install -y build-essential &&     pip install --upgrade pip setuptools
  => CACHE [4/6] COPY requirements.txt .
  => CACHE [5/6] RUN pip install --no-cache-dir -r requirements.txt
  => CACHE [6/6] COPY .
  => exporting to image
  => exporting layers
  => writing image sha256:72ef1e7badbf4fb9a1476691a6481ec9754843dc93b75e3a24d7c8e0aafdf9091
  => naming to docker.io/library/prog1
1RV24MC063_mahadevprasad_dl@girish-HP-Laptop-15s-fr5xxx:~/Dockerfile/prog1$ sudo docker run -it -p 5000:5000 prog1
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.18.0.2:5000
Press CTRL+C to quit
172.18.0.1 - - [30/Oct/2025 15:53:36] "GET / HTTP/1.1" 200 -
```