

Docker Compose week 6

1. Define and run multiple interdependent services

Word Press:

```
version: '3.8' # Docker Compose file format version

services:
  wordpress: # WordPress service
    image: wordpress:latest
    ports:
      - "8236:80" # Map port 80 of the container to port 8080 of the host
    environment:
      WORDPRESS_DB_HOST: db:3306 # Database host
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    depends_on:
      - db # Ensures the db service starts first

  db: # MySQL service
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
```

1. To Start the compose

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose -f compose.yaml up -d
time="2025-08-27T09:56:58+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2025-08-27T09:56:58+05:30" level=warning msg="Found orphan containers ([dockercospose-wordpress-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 2/2
  ✓ Container dockercompose-web-1    started          0.7s
  ✓ Container dockercompose-db-1    started          0.5s
```

2. To stop the containers

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose down
time="2025-08-27T09:58:52+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
  ✓ Container dockercompose-web-1  Removed          0.5s
  ✓ Container dockercompose-db-1  Removed          0.4s
  ✓ Network dockercompose_default Removed          0.3s
```

3.To scale the container

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker-compose up --scale db=2 -d
time="2025-08-27T09:59:57+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2025-08-27T09:59:57+05:30" level=warning msg="Found orphan containers ([dockercompose-wordpress-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 4/4
  ✓ Network dockercompose_default  Created          0.0s
  ✓ Container dockercompose-db-2  Started          1.1s
  ✓ Container dockercompose-web-1 Started          0.8s
  ✓ Container dockercompose-db-1  Started          0.8s
```

To check the network

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
de0b7aef9ab2    bridge    bridge      local
939e68639102    dockercompose_default bridge      local
7887a166bcb2    host      host       local
a0eb77da1f1e    none      null       local
```

create a file docker-compose.yml

```
❸ compose.yaml
1   version: "3.9"
2   services:
3     web:
4       image: nginx:latest
5       ports:
6         - "8088:80"
7
8     db:
9       image: postgres:15
10      environment:
11        POSTGRES_USER: demo
12        POSTGRES_PASSWORD: demo
13        POSTGRES_DB: demo_db
14
15
```

Run the setup

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose -f compose.yaml up -d
time="2025-08-26T18:04:33+05:30" level=warning msg="C:\\Users\\azeem\\OneDrive\\Desktop\\Dockercompose\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2025-08-26T18:04:33+05:30" level=warning msg="Found orphan containers ([dockercompose-wordpress-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 2/2
  ✓ Container dockercompose-db-1  Running          0.0s
  ✓ Container dockercompose-web-1 Started          1.5s
```

Open your browser and visit: <http://localhost:8088>

localhost:8088

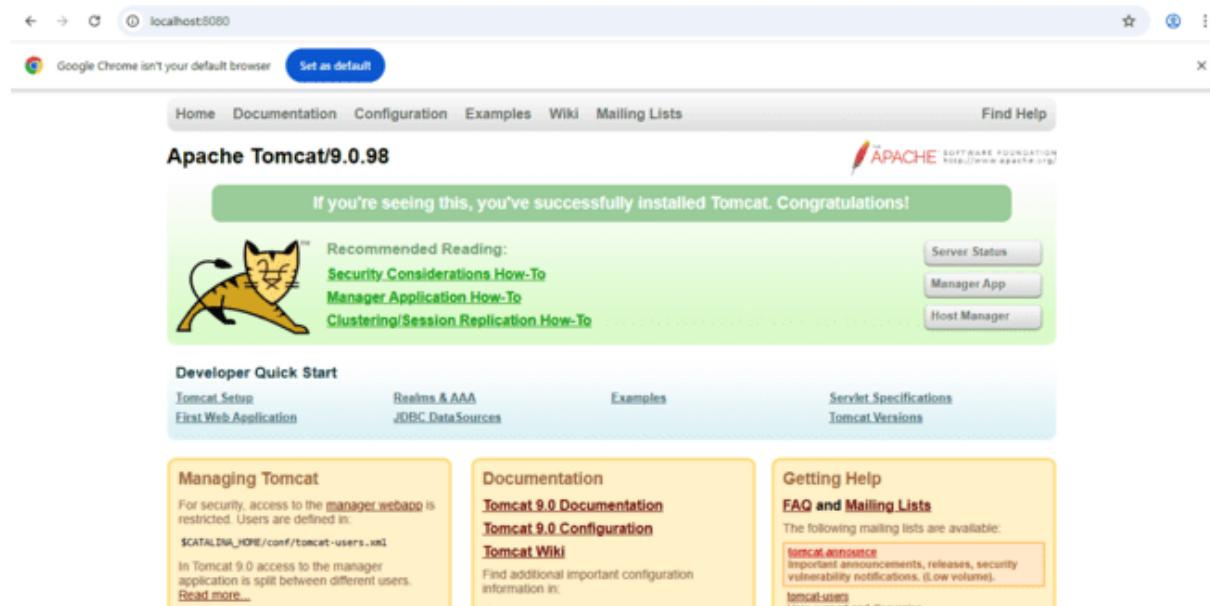
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Changed the default port because the Tomcat server is already running on the default port 8080



To stop the containers

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker-compose down
time="2025-08-27T09:37:11+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/2
  ✓ Container dockercompose-db-1   Removed          0.8s
  ✓ Container dockercompose-web-1  Removed          0.8s
  ! Network dockercompose_default  Resource is sti...          0.0s
```

2. Write and interpret docker-compose.yml files

I. Modify docker-compose.yml to add a Redis cache:

```

👉 compose.yaml
1   version: "3.9"
2   services:
3     web:
4       image: nginx:latest
5       ports:
6         - "8088:80"
7
8     db:
9       image: postgres:15
10      environment:
11        POSTGRES_USER: demo
12        POSTGRES_PASSWORD: demo
13        POSTGRES_DB: demo_db
14

```

Restart the setup:

```

azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose -f compose.yaml up -d
time="2025-08-27T09:39:29+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2025-08-27T09:39:29+05:30" level=warning msg="Found orphan containers ([dockercompose-wordpress-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 2/2
  ✓ Container dockercompose-web-1  Started                               1.0s
  ✓ Container dockercompose-db-1  Started                               1.0s

```

Docker compose ps

```

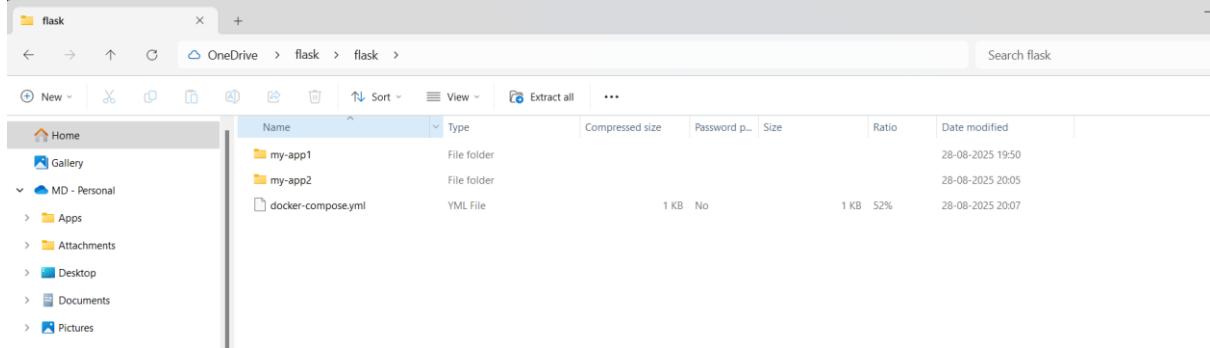
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose ps
time="2025-08-27T09:40:05+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
NAME          STATUS    PORTS           COMMAND          SERVICE      CREATE
Dockercompose-db-1  Up 36 seconds  postgres:15  "docker-entrypoint.s..."  db  37 sec
on 36 seconds  5432/tcp
Dockercompose-web-1  Up 36 seconds  nginx:latest  "/docker-entrypoint..."  web  37 sec
on 36 seconds  0.0.0.0:8088->80/tcp
Dockercompose-wordpress-1  Up 16 hours  wordpress:latest  "docker-entrypoint.s..."  wordpress  16 hours
on 16 hours  0.0.0.0:8086->80/tcp

```

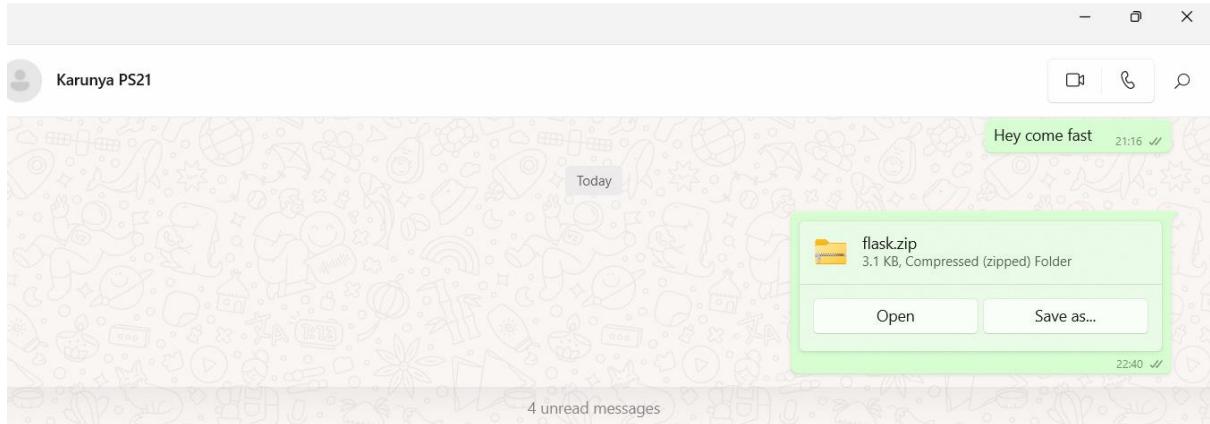
Three services (web, db, redis) are listed as running.

Task: 3 Deploy across different machines

Zip your compose-lab folder.



Shared the zip file to my friend



II. Run:

docker compose up -d

```
daman@karunya MINGW64 ~/OneDrive/Documents/Flask[1]/Flask (main)
$ ls
docker-compose.yml  my-app1/  my-app2/
daman@karunya MINGW64 ~/OneDrive/Documents/Flask[1]/Flask (main)
$ docker --version
Docker version 28.3.2, build 578ccf6
daman@karunya MINGW64 ~/OneDrive/Documents/Flask[1]/Flask (main)
[+] Running 0/0
✓ db Pulled
✓ a0ef231c0fc5 Pull complete
✓ a3be5d4ce401 Pull complete
✓ d403fe017deaa Pull complete
✓ 81a4f01c8a98 Pull complete
✓ 6dc486ace974 Pull complete
✓ b0e3ae2a5b88 Pull complete
✓ d691d168572b Pull complete
✓ e27d17405276 Pull complete
#1 [internal] load local bake definitions
#1 reading from stdin 1.08kB done
#1 DONE 0.0s

#2 [backend internal] load build definition from Dockerfile
#2 transferring dockerfile: 320B 0.0s done
#2 DONE 0.1s

#3 [frontend internal] load build definition from Dockerfile
#3 transferring dockerfile: 320B 0.0s done
#3 DONE 0.1s

#4 [frontend internal] load metadata for docker.io/library/node:18
#4 DONE 5.4s

#5 [frontend internal] load .dockerrcignore
#5 transferring context: 2B done
#5 DONE 0.1s

#6 [backend internal] load .dockerrcignore
#6 transferring context: 2B done
#6 DONE 0.1s

#7 [backend internal] load build context
#7 transferring context: 818B done
#7 DONE 0.1s

#8 [frontend internal] load build context
#8 transferring context: 858B done
#8 DONE 0.1s

#9 [frontend 1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
#9 resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783 0.0s done
#9 sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145 0B / 1.25MB 0.2s
#9 sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 0B / 211.36MB 0.2s
#9 sha256:cda7f4af2bddc4bb7514474024b3f3705de00dd6355a33be5ac7808e5b7125 0B / 3.32KB 0.2s
#9 sha256:461077a72fb7fe40d34a37d6a1958c4d16772d0dd77f572ec50a1fdc41a3754d 0B / 446B 0.2s
```

```

#9 extracting sha256:79b2f4/ad4443652b9b5cc81a95e0e249fd976310efdb00159f29638783778c0
#9 extracting sha256:37927ed901b1b2608b72796c6881bf645480268eca4ac9a37b9219e050bb4d84 0.8s done
#9 extracting sha256:79b2f47ad4443652b9b5cc81a95e0e249fd976310efdb00159f29638783778c0 1.8s done
#9 extracting sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab
#9 extracting sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 5.1s done
#9 extracting sha256:e23f099911d692f62b851cf49a1e93294288a115f5cd2d014180e4d3684d34ab 5.1s done
#9 extracting sha256:cda7f44f2bddcc4bb7514474024b3f3705de00ddb6355a33be5ac7808e5b7125 0.0s done
#9 extracting sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7
#9 extracting sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7 1.5s done
#9 extracting sha256:c6b30c3f16966552af10ac00521f60355b1fcfd46ac1c20b1038587e28583ce7 1.5s done
#9 extracting sha256:3697be50c98b9d071df4637e1d3491d00e7b9f3a732768c876d82309b3c5a145 0.1s done
#9 extracting sha256:461077a72fb7fe40d34a37d0a1958c4d16772d0dd77f572ec50a1fdc41a3754d 0.0s done
#9 DONE 29.8s

#10 [backend 2/5] WORKDIR /usr/src/app
#10 DONE 1.9s

#11 [backend 3/5] COPY package*.json .
#11 DONE 0.1s

#12 [frontend 3/5] COPY package*.json .
#12 DONE 0.2s

#13 [frontend 4/5] RUN npm install
#13 4.985
#13 4.985 added 69 packages, and audited 70 packages in 4s
#13 4.985
#13 4.985 14 packages are looking for funding
#13 4.985 run `npm fund` for details
#13 4.986
#13 4.986 found 0 vulnerabilities
#13 4.988 npm notice
#13 4.988 npm notice New major version of npm available! 10.8.2 -> 11.5.2
#13 4.988 npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
#13 4.988 npm notice To update run: npm install -g npm@11.5.2
#13 4.988 npm notice
#13 ...

#14 [backend 4/5] RUN npm install
#14 5.010
#14 5.010 added 69 packages, and audited 70 packages in 4s
#14 5.010
#14 5.010 14 packages are looking for funding
#14 5.010 run `npm fund` for details
#14 5.011
#14 5.011 found 0 vulnerabilities
#14 5.013 npm notice
#14 5.013 npm notice New major version of npm available! 10.8.2 -> 11.5.2
#14 5.013 npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
#14 5.013 npm notice To update run: npm install -g npm@11.5.2
#14 5.013 npm notice
#14 DONE 5.6s

#13 [frontend 4/5] RUN npm install
#13 DONE 5.6s

#15 DONE 0.2s

#17 [frontend] exporting to image
#17 exporting layers
#17 exporting layers 0.9s done
#17 exporting manifest sha256:3a1c5ff44fb0050e7f7749faa8e235b7397f01681b91ae3b14791c0a2b15f038 0.1s done
#17 exporting config sha256:8ee37251ca334b50f8b5395ce6fd88e13c5eb2c5588d9207f67ecd0d44d5b230 0.1s done
#17 exporting attestation manifest sha256:f8117caf2a25c1b76e2d75ada8dec3aa53bf7b1c7b11ca8edd16c27c7924690f
#17 exporting attestation manifest sha256:f8117caf2a25c1b76e2d75ada8dec3aa53bf7b1c7b11ca8edd16c27c7924690f 0.1s done
#17 exporting manifest list sha256:ee8669247c511823dbc5ff0a566e5131ebf1bbec0c724dc89d52e82a/29ca335 0.0s done
#17 naming to docker.io/library/flask-frontend:latest done
#17 unpacking to docker.io/library/flask-frontend:latest
#17 unpacking to docker.io/library/flask-frontend:latest 0.6s done
#17 DONE 1.9s

#18 [backend] exporting to image
#18 exporting layers 0.9s done
#18 exporting manifest sha256:78ae900dbd16b695862d9f47415002c17435fb702fdfa3444fbfc6d240a569c2 0.0s done
#18 exporting config sha256:1062d56d35ebdb849bff3a72d542d57e7c80a41e3d642eb496b3b14bcd49bfe 0.1s done
#18 exporting attestation manifest sha256:29ff05369126cad7917252bb1f43da6ebc123d976700eee278123ab994d38b 0.1s done
#18 exporting manifest list sha256:c47bfc0420dbc3ce0471052726951ab906399fdafce9c729c7740c91eb59d76 0.0s done
#18 naming to docker.io/library/flask-backend:latest 0.0s done
#18 unpacking to docker.io/library/flask-backend:latest 0.6s done
#18 DONE 1.9s

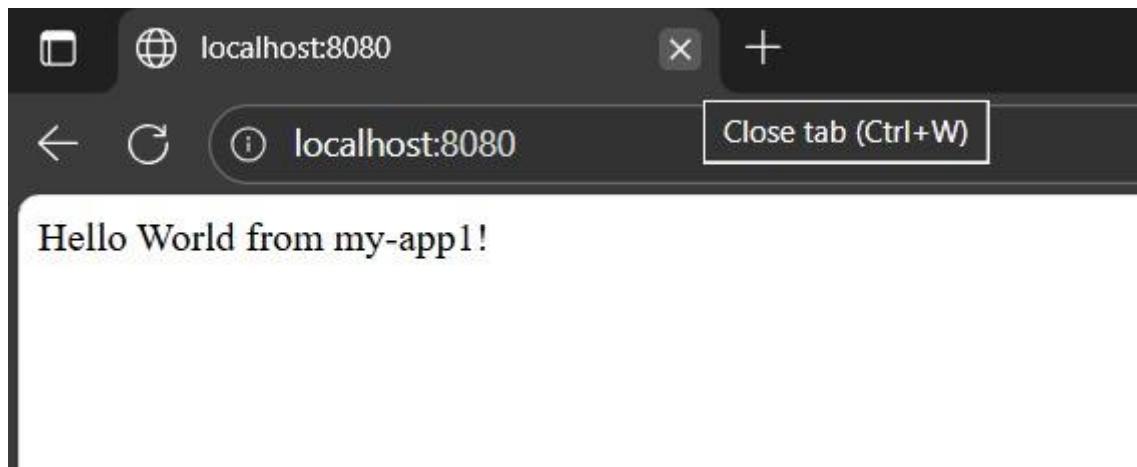
#19 [backend] resolving provenance for metadata file
#19 DONE 0.1s

#20 [frontend] resolving provenance for metadata file
#20 DONE 0.0s
[+] Running 7/7
  ✓ flask-frontend          Built
  ✓ flask-backend            Built
  ✓ Network flask_default   Created
  ✓ Volume "flask_mongo-data" Created
  ✓ Container flask-backend-1 Started
  ✓ Container my-mongo      Started
  ✓ Container flask-frontend-1 Started

bamana@karunya MINGW64 ~/OneDrive/Documents/flask[1]/flask (main)
$ 

```

The same services run on the new machine without changes.



Task:4 Networking and persistent storage

- I. **Update your docker-compose.yml to add a custom network and volume:**

```

compose.yaml
services:
  web:
    image: nginx:latest
    ports:
      - "8088:80"
    networks:
      - app-net
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
      POSTGRES_USER: demo
      POSTGRES_PASSWORD: demo
      POSTGRES_DB: demo_db
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - app-net

networks:
  app-net:

volumes:
  db-data:

```

2. Run:

```

azeem@LAPTOP-MUV4INQ MINGW64 ~/onedrive/Desktop/Dockercompose (main)
$ docker compose -f compose.yaml up -d
[+] Running 2/2
  ✓ Network dockercompose_app-net  Created
  ✓ Volume "dockercompose_db-data" Created
[+] Running 5/5 | 10:05:06+05:30" Level=warning msg="Found orphan containers ([docke...
  ✓ Network dockercompose_app-net  Created
  ✓ Volume "dockercompose_db-data" Created
  ✓ Container dockercompose-web-1 Started
  ✓ Container dockercompose-db-2  Removed
  ✓ Container dockercompose-db-1 Started

```

3.Remove containers:

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose -f compose.yaml down
[+] Running 3/3
  ✓ Container dockercompose-web-1   Removed
  ✓ Container dockercompose-db-1   Removed
  ✓ Network dockercompose_app-net  Removed
```

4. Start again:

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose -f compose.yaml up -d
[+] Running 1/0
  ✓ Network dockercompose_app-net  Created          0.0s
time="2025-08-27T10:06:32+05:30" level=warning msg="Found orphan containers ([docke
[+] Running 3/3 project. If you removed or renamed this service in your compose file, you can run t
  ✓ Network dockercompose_app-net  Created          0.0s
  ✓ Container dockercompose-db-1  Started          0.6s
  ✓ Container dockercompose-web-1 Started          0.8s
```

Database data persists across restarts.

Services communicate via the app-net network using service names.

Task 5: Faster iteration during development

I. Create a simple Flask app in app.py:

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

II. Add a Dockerfile in the same folder:

```
↳ dockerfile
1  FROM python:3.10-slim
2  WORKDIR /app
3  COPY app.py /app/
4  RUN pip install flask
5  CMD ["python", "app.py"]
6
```

III. Update docker-compose.yml:

```
mpose.yaml
  services:
    web:
      build: .
      ports:
        - "5852:5000"
      depends_on:
        - db

    db:
      image: postgres:13
      environment:
        POSTGRES_USER: user
        POSTGRES_PASSWORD: password
        POSTGRES_DB: mydb
      ports:
        - "5432:5432"
```

```

azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose up --build -d
time="2025-08-27T10:37:43+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 0/1
[+] Building 1.4s (9/9) FINISHED
=> [web internal] load build definition from dockerfile docker:desktop-linux
=> => transferring dockerfile: 142B docker:desktop-linux
=> [web internal] load metadata for docker.io/library/python:3.10-slim 0.0s
=> [web internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [web 1/4] FROM docker.io/library/python:3.10-slim@sha256:420fbb0e468d3eaf0f7e93ea6f7a4874 0.0s
=> => resolve docker.io/library/python:3.10-slim@sha256:420fbb0e468d3eaf0f7e93ea6f7a48792cb 0.0s
=> [web internal] load build context 0.0s
=> => transferring context: 28B 0.0s
=> CACHED [web 2/4] WORKDIR /app 0.0s
=> CACHED [web 3/4] COPY app.py /app/ 0.0s
=> CACHED [web 4/4] RUN pip install flask 0.0s
=> [web] exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:1a742e950611dc790ff77d20d7854e06870f4ad3bd1152250bb932f4874 0.0s
=> => exporting config sha256:890daf88092ebc3ec428b9c6626771b86a92ab056e164706b9b1e4dd4af53 0.0s
=> => exporting attestation manifest sha256:91d9c668bb65f673de48ea9d2d5b5961b4c9ebd1c567bd 0.0s
=> => exporting manifest list sha256:47af7c9fc2cb2b9d1dea520ff8d51ee8ea7e6ea02390d3e7e3944 0.0s
=> => naming to docker.io/library/dockercompose-web:latest 0.0s
[+] Running 3/3g to docker.io/library/dockercompose-web:latest 0.0s

```

IV. Run:

docker compose up --build

```

azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/Dockercompose (main)
$ docker compose up --build
time="2025-08-27T10:43:42+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\OneDrive\\\\Desktop\\\\Dockercompose\\\\compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
=> [web internal] load metadata for docker.io/library/python:3.10-slim 1.7s
=> [web internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [web 1/4] FROM docker.io/library/python:3.10-slim@sha256:420fbb0e468d3eaf0f7e93ea6f7a4874 0.0s
=> => resolve docker.io/library/python:3.10-slim@sha256:420fbb0e468d3eaf0f7e93ea6f7a48792cb 0.0s
=> [web internal] load build context 0.0s
=> => transferring context: 28B 0.0s
=> CACHED [web 2/4] WORKDIR /app 0.0s
=> CACHED [web 3/4] COPY app.py /app/ 0.0s
=> CACHED [web 4/4] RUN pip install flask 0.0s
=> [web] exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:b7b637683cfcd3644f280a82c98bdb0b887e338ca7a4b6c13751a1ea9e 0.0s
=> => exporting config sha256:062ec732b001192bc0050c18a72693236d5d2bd52b3971719f6b51d576d0 0.0s
=> => exporting attestation manifest sha256:f9d975e04c3cdc7f654e5283cf21c783d4238a169103f01 0.0s
=> => exporting manifest list sha256:c93a05b4f03d384c81ca6d49bf58df27b7fffb0b16d616be7d15bc 0.0s
=> => naming to docker.io/library/dockercompose-web:latest 0.0s
[+] Running 3/3g to docker.io/library/dockercompose-web:latest 0.0s
✓ Service web Built 2.4s
✓ Container dockercompose-db-1 Running 0.0s
✓ Container dockercompose-web-1 Recreated 10.5s
Attaching to db-1, web-1
web-1 | Traceback (most recent call last):
web-1 |   File "/app/app.py", line 10, in <module>
web-1 |     import pymongo
web-1 | ModuleNotFoundError: No module named 'pymongo'
web-1 exited with code 1

```



Task 6: Scenario based Questions:

1. You have two applications — a Node.js backend and a Python script that handles scheduled jobs. You're considering running both inside the same container. Should you run multiple applications in the same Docker container?

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ nano compose5.yml
```

```
MINGW64/c/Users/azeem/OneDrive/Desktop/flask
GNU nano 8.5
compose5.yml
version: "3"
services:
  backend:
    build: ./backend # Node.js
    ports:
      - "3000:3000"
  scheduler:
    build: ./scheduler # Python script

[ Read 9 lines ]
```

```

MINGW64:/c/Users/azeem/OneDrive/Desktop/flask
azeem@LAPTOP-MUVDAING MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker compose up -d --build
[+] Running 0/2nd Building
--> transferring dockerfile: 320B
[+] Building 4.7s (20/20) FINISHED
--> [frontend internal] load build definition from Dockerfile 0.1s
--> transferring dockerfile: 320B 0.1s
--> [backend internal] load build definition from Dockerfile 0.1s
--> transferring dockerfile: 320B 0.1s
--> [backend internal] load metadata for docker.io/library/node:18 0.0s
--> [frontend auth] library/node:pull token for registry-1.docker.io 0.0s
--> [backend internal] load .dockerignore 0.0s
--> transferring context: 28 0.0s
--> [frontend internal] load .dockerignore 0.0s
--> transferring context: 28 0.0s
--> [backend 1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d94349855 0.1s
--> => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d378 0.0s
--> [backend internal] load build context 0.0s
--> => transferring context: 90B 0.0s
--> [frontend internal] load build context 0.0s
--> => transferring context: 90B 0.0s
--> [backend 2/5] WORKDIR /usr/src/app 0.0s
--> CACHED [frontend 3/5] COPY package*.json ./ 0.0s
--> CACHED [frontend 4/5] RUN npm install 0.0s
--> CACHED [frontend 5/5] COPY . 0.0s
--> CACHED [backend 3/5] COPY package*.json ./ 0.0s
--> CACHED [backend 4/5] RUN npm install 0.0s
--> CACHED [backend 5/5] COPY . 0.0s
--> [frontend] exporting to image 0.2s
--> => exporting layers 0.0s
--> => exporting manifest sha256:d5a0c552bd39e2c4b9827ce8313269e2bc2f5dcc32aedfc7190dae8e7bf0b0 0.0s
--> => exporting config sha256:8b2d40b0024d3fd5dc76da31349673875d93ab8a44630912fcba9fe5831a2ab 0.0s
--> => exporting attestation manifest sha256:3eededac590033b4241e55544fdbaf8e5325c3b42e8f75af52ed8f5a0624539c 0.1s
--> => exporting manifest list sha256:1ff57e5d7c5df65add06c23dea8518bf6228dab22e4c7b1d0674e6a0d2df8c1 0.0s
--> => naming to docker.io/library/flask-frontend:latest 0.0s
--> => unpacking to docker.io/library/flask-frontend:latest 0.0s
--> [backend] exporting to image 0.2s
--> => exporting layers 0.0s

```

We can use docker compose

```

MINGW64:/c/Users/azeem/OneDrive/Desktop/flask
azeem@LAPTOP-MUVDAING MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker compose up -d
--> [frontend internal] load .dockerignore 0.0s
--> transferring context: 28 0.0s
--> [backend 1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d94349855 0.1s
--> => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d378 0.0s
--> [backend internal] load build context 0.0s
--> => transferring context: 90B 0.0s
--> [frontend internal] load build context 0.0s
--> => transferring context: 90B 0.0s
--> [backend 2/5] WORKDIR /usr/src/app 0.0s
--> CACHED [frontend 3/5] COPY package*.json ./ 0.0s
--> CACHED [frontend 4/5] RUN npm install 0.0s
--> CACHED [frontend 5/5] COPY . 0.0s
--> CACHED [backend 3/5] COPY package*.json ./ 0.0s
--> CACHED [backend 4/5] RUN npm install 0.0s
--> CACHED [backend 5/5] COPY . 0.0s
--> [frontend] exporting to image 0.2s
--> => exporting layers 0.0s
--> => exporting manifest sha256:d5a0c552bd39e2c4b9827ce8313269e2bc2f5dcc32aedfc7190dae8e7bf0b0 0.0s
--> => exporting config sha256:8b2d40b0024d3fd5dc76da31349673875d93ab8a44630912fcba9fe5831a2ab 0.0s
--> => exporting attestation manifest sha256:3eededac590033b4241e55544fdbaf8e5325c3b42e8f75af52ed8f5a0624539c 0.1s
--> => exporting manifest list sha256:1ff57e5d7c5df65add06c23dea8518bf6228dab22e4c7b1d0674e6a0d2df8c1 0.0s
--> => naming to docker.io/library/flask-frontend:latest 0.0s
--> => unpacking to docker.io/library/flask-frontend:latest 0.0s
--> [backend] exporting to image 0.2s
--> => exporting layers 0.0s
--> => exporting manifest sha256:c78927f3d50f4078936b4c69a6a02748e204c6c9401d9cf0adaef0cb91c795a 0.0s
--> => exporting config sha256:a77ae65635d93ff94bf3377209b1l1dca06b178f4d4c939a7a0203579511d5dd 0.0s
--> => exporting attestation manifest sha256:4fe239f073fb29c3435115e3bc649b33927b995c21adbc757fa989df66202a7 0.1s
--> => exporting manifest list sha256:b2c3886874b66e7a454c271be27923bf80dcb6eaf81012bddab9a11f906f7c89 0.0s
--> => naming to docker.io/library/flask-backend:latest 0.0s
[+] Running 5/5g to docker.io/library/flask-backend:latest 0.0s
✓ Service frontend Built 5.8s
✓ Service backend Built 5.8s
✓ Container my-mongo Running 0.0s
✓ Container flask-backend-1 Started 2.4s
✓ Container flask-frontend-1 Started 2.4s

azeem@LAPTOP-MUVDAING MINGW64 ~/OneDrive/Desktop/flask (main)
$ |

```

2. You have a Flask API and an Nginx server. You want to run both containers on the same host and expose them on ports 5000 and 80 respectively. How can you expose both applications from different containers?

```

GNU nano 8.5                               compose1.yml                                Modified
services:
  flask_api:
    build: ./backend
    container_name: flask_api
    ports:
      - "5000:5000"
    networks:
      - mynetwork

  nginx:
    image: nginx:latest
    container_name: nginx_server
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - flask_api
    networks:
      - mynetwork

networks:
  mynetwork:
    driver: bridge

```

The screenshot shows a terminal window titled 'MINGW64/c/Users/azeem/OneDrive/Desktop/flask'. Inside the terminal, a file named 'compose1.yml' is being edited with the nano text editor. The file contains a Docker Compose configuration for a Flask API and an Nginx server, both running on a network named 'mynetwork'. The terminal window has a standard Windows-style menu bar at the top and a taskbar with various icons at the bottom.

3. You want to run a React frontend, an Express.js backend and MongoDB – all together.

How do you run and manage all three together?

```

GNU nano 8.5                               compose2.yml                                Modified
version: "3.8"
services:
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
    depends_on:
      - backend

  backend:
    build: ./backend
    ports:
      - "5000:5000"
    depends_on:
      - mongo
    environment:
      - MONGO_URL=mongodb://mongo:27017/mydb

  mongo:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:

```

The screenshot shows a terminal window titled 'MINGW64/c/Users/azeem/OneDrive/Desktop/flask'. Inside the terminal, a file named 'compose2.yml' is being edited with the nano text editor. This configuration uses version 3.8 of Docker Compose and defines three services: 'frontend', 'backend', and 'mongo'. The 'frontend' service runs on port 3000. The 'backend' service runs on port 5000 and depends on the 'mongo' service. The 'mongo' service runs on port 27017. A volume named 'mongo-data' is mounted at /data/db. The terminal window has a standard Windows-style menu bar at the top and a taskbar with various icons at the bottom.

```

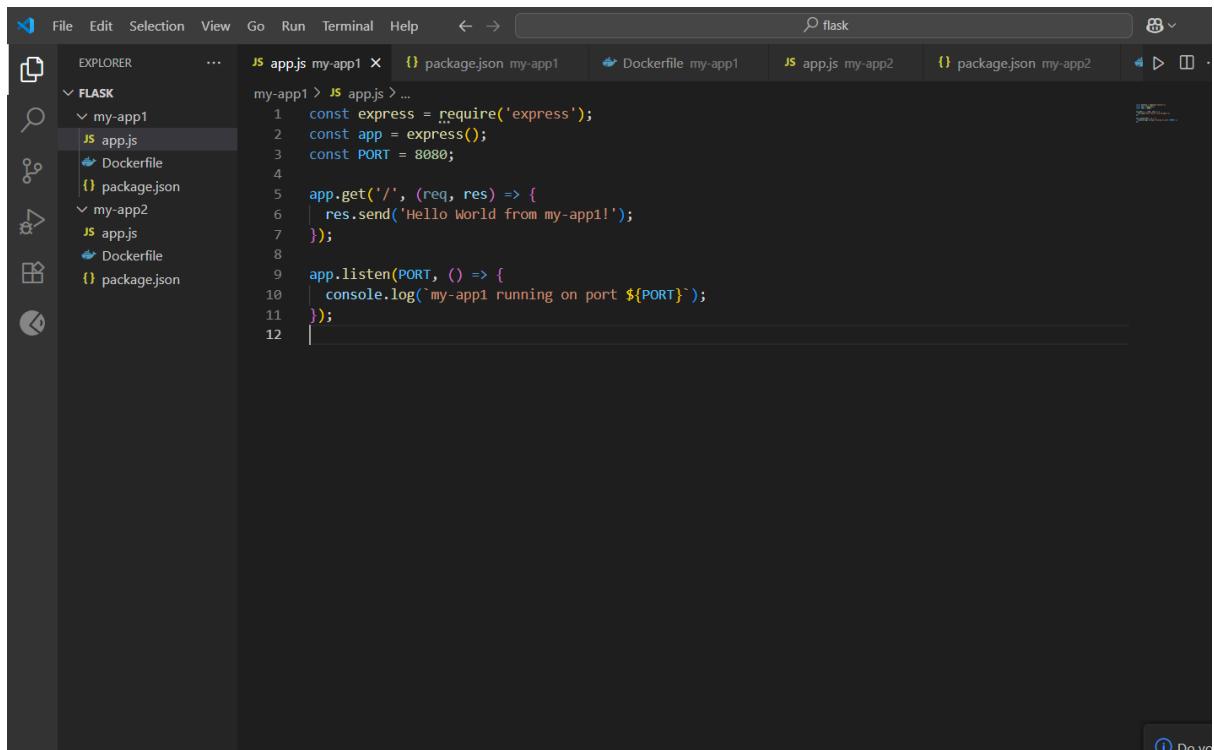
azeem@LAPTOP-MUV04INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker compose -f compose2.yml up -d
time="2025-08-28T22:58:02+05:30" level=warning msg="C:\\\\Users\\\\azeem\\\\onedrive\\\\Desktop\\\\flask\\\\compose2.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
time="2025-08-28T22:58:02+05:30" level=warning msg="Found orphan containers ([my-mongo]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up."
[+] Running 2/3
- Container flask-mongo-1 Starting                                         2.9s
✓ Container flask-frontend-1 Recreated                                       2.5s
✓ Container flask-backend-1 Recreated                                         1.3s

```

The screenshot shows a terminal window with the command 'docker compose -f compose2.yml up -d' being run. The output shows several warning messages related to deprecated Docker Compose features like 'version' and 'my-mongo'. It also indicates that orphan containers are found and suggests using the '--remove-orphans' flag. The command then proceeds to start three containers: 'flask-mongo-1', 'flask-frontend-1', and 'flask-backend-1'. The terminal window has a standard Windows-style menu bar at the top and a taskbar with various icons at the bottom.

4. You try to run two containers that both expose port 8080 on the host.

What happens, and how can you run both apps?



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** flask
- Explorer View:** Shows two projects:
 - my-app1:** Contains files: app.js, Dockerfile, package.json.
 - my-app2:** Contains files: app.js, Dockerfile, package.json.
- Code Editor:** The app.js file for my-app1 is open, displaying the following code:

```
const express = require('express');
const app = express();
const PORT = 8080;

app.get('/', (req, res) => {
  res.send('Hello World from my-app1!');
});

app.listen(PORT, () => {
  console.log(`my-app1 running on port ${PORT}`);
});
```

The screenshot shows two separate instances of Visual Studio Code running side-by-side. Both instances have dark themes and are displaying Node.js projects.

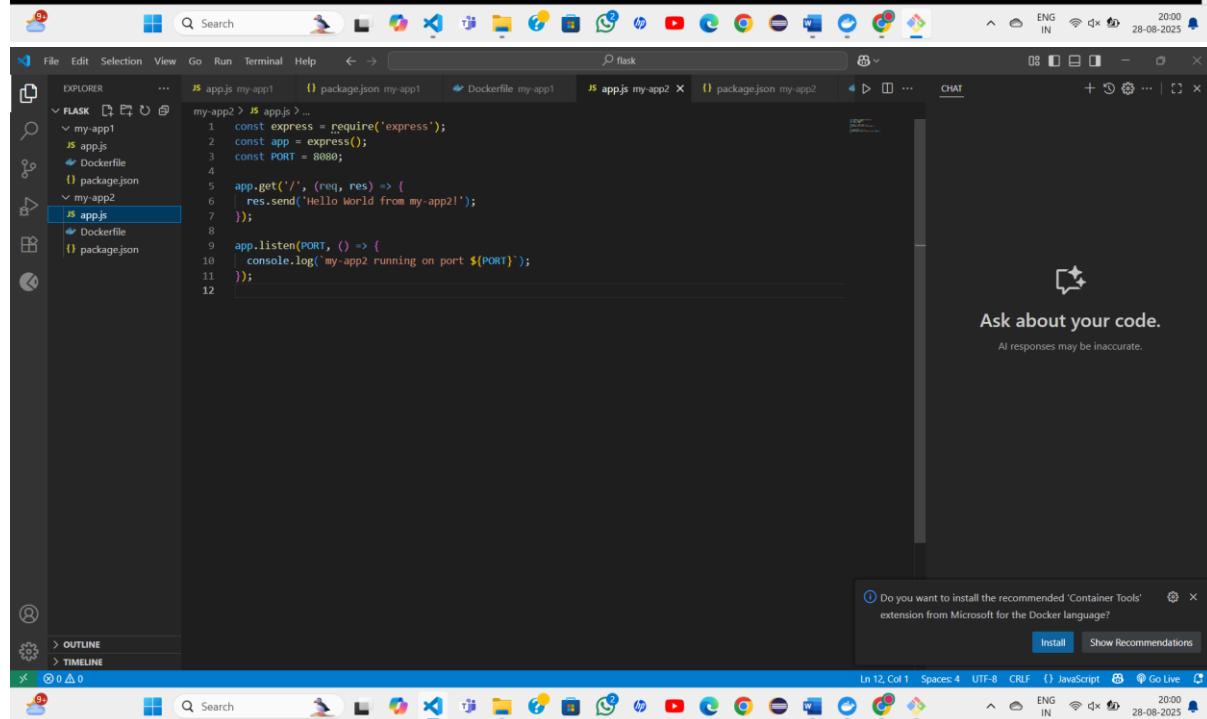
Top Instance (Left):

- Explorer:** Shows a FLASK folder containing my-app1 and my-app2. my-app1 contains app.js, Dockerfile, and package.json. my-app2 contains app.js, Dockerfile, and package.json.
- Dockerfile (my-app1):**

```
my-app1 > Dockerfile
1 # Use Node.js base image
2 FROM node:18
3
4 # Set working directory
5 WORKDIR /usr/src/app
6
7 # Copy package files
8 COPY package*.json .
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy app files
14 COPY .
15
16 # Expose port 8080
17 EXPOSE 8080
18
19 # Start app
20 CMD [ "npm", "start" ]
21
```
- Bottom Instance (Right):**
- Explorer:** Shows a FLASK folder containing my-app1 and my-app2. my-app1 contains app.js, Dockerfile, and package.json. my-app2 contains app.js, Dockerfile, and package.json.
- Dockerfile (my-app1):**

```
my-app1 > Dockerfile
1 {
2   "name": "my-app1",
3   "version": "1.0.0",
4   "description": "simple Node.js app",
5   "main": "app.js",
6   "scripts": {
7     "start": "node app.js"
8   },
9   "dependencies": {
10     "express": "4.18.2"
11   }
12 }
```
- Common UI Elements:** Both instances share a top navigation bar with File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. They also have a status bar at the bottom showing file paths, line numbers, spaces, encoding, and date.
- Extension Pop-up:** A tooltip message appears in both instances: "Do you want to install the recommended 'Container Tools' extension from Microsoft for the Docker language?". It includes "Install" and "Show Recommendations" buttons.

```
MINGW64:/c/Users/azeem/OneDrive/Desktop/flask (main)
$ docker build -t my-app1 ./my-app1
[+] Building 10.2s (10/10) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 320B                                         0.0s
=> [internal] load metadata for docker.io/library/node:18                   1.3s
=> [internal] load .dockerignore                                           0.0s
=> => transferring context: 2B                                           0.0s
=> [1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e 0.1s
=> => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e 0.1s
=> [internal] load build context                                           0.0s
=> => transferring context: 315B                                         0.0s
=> CACHED [2/5] WORKDIR /usr/src/app                                       0.0s
=> [3/5] COPY package*.json ./                                              0.0s
=> [4/5] RUN npm install                                                 6.6s
=> [5/5] COPY . .                                                       0.1s
=> exporting to image                                                 1.7s
=> => exporting layers                                                 0.8s
=> => exporting manifest sha256:10079a0cbbfe0797384b553a1ae7f9df41e34a73e8 0.0s
=> => exporting config sha256:0541dd220f540fec8b601ade8eb143af6fe3d4fa51cb 0.0s
=> => exporting attestation manifest sha256:52608545ad483e593d2d1e4baa414b 0.0s
=> => exporting manifest list sha256:75241b323330d85d0d0914f09d39d5d4c94ab 0.0s
=> => naming to docker.io/library/my-app1:latest                           0.0s
=> => unpacking to docker.io/library/my-app1:latest                         0.7s
```



The screenshot displays two instances of Visual Studio Code (VS Code) running side-by-side. Both instances have dark themes.

Top Window:

- Explorer:** Shows a project structure under 'FLASK'. It includes 'my-app1' (with 'app.js' and 'Dockerfile'), 'my-app2' (with 'app.js' and 'Dockerfile'), and a 'package.json' file.
- Editor:** Displays the contents of the 'Dockerfile' for 'my-app2'. The Dockerfile specifies a Node.js base image, sets the working directory to /usr/src/app, copies package files, installs dependencies, copies app files, exposes port 8080, and starts the app with npm.
- Bottom Status Bar:** Shows 'Ln 21, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Docker', 'Go Live', and system status including ENG IN, 20:00, and 28-08-2025.

Bottom Window:

- Explorer:** Shows a project structure under 'FLASK'. It includes 'my-app1' (with 'app.js' and 'Dockerfile'), 'my-app2' (with 'app.js' and 'Dockerfile'), and a 'package.json' file.
- Editor:** Displays the contents of the 'package.json' file for 'my-app2'. It defines the name as 'my-app2', version as '1.0.0', main as 'app.js', and scripts with a start command.
- Bottom Status Bar:** Shows 'Ln 12, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'JSON', 'Docker', 'Go Live', and system status including ENG IN, 20:00, and 28-08-2025.

In both windows, there is a floating 'Ask about your code.' AI interface in the bottom right corner. A tooltip at the bottom of the interface asks if the user wants to install the 'Container Tools' extension from Microsoft for the Docker language.

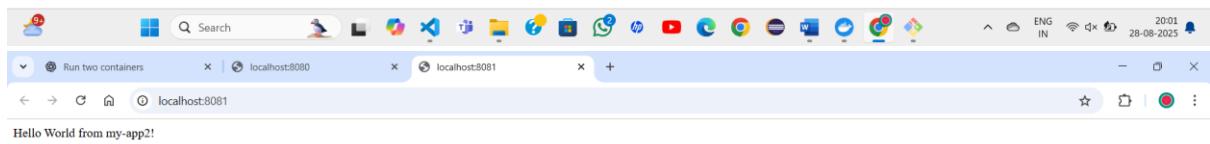
```
MINGW64:/c/Users/azeem/OneDrive/Desktop/flask (main)
$ docker build -t my-app2 ./my-app2
[+] Building 9.4s (10/10) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                      0.1s
=> => transferring dockerfile: 320B                                         0.0s
=> [internal] load metadata for docker.io/library/node:18                  0.7s
=> [internal] load .dockignore                                              0.1s
=> => transferring context: 2B                                           0.0s
=> [1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e 0.1s
=> => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e 0.1s
=> [internal] load build context                                           0.1s
=> => transferring context: 818B                                         0.0s
=> CACHED [2/5] WORKDIR /usr/src/app                                       0.0s
=> [3/5] COPY package*.json ./                                              0.1s
=> [4/5] RUN npm install                                                 6.3s
=> [5/5] COPY . .                                                       0.1s
=> exporting to image                                                 1.7s
=> => exporting layers                                              0.8s
=> => exporting manifest sha256:9e3ca47834c1027f0f36606ee6016d2ac6dfc47ed8 0.0s
=> => exporting config sha256:b80c9d2675ad4beb7321a0c3e3b31959f4e92ae5604f 0.0s
=> => exporting attestation manifest sha256:65afc6a7d4c581773ec7caefca0a70 0.0s
=> => exporting manifest list sha256:95a3c2310764473421ace0d566263bb427b4 0.0s
=> => naming to docker.io/library/my-app2:latest                         0.0s
```

```
MINGW64:/c/Users/azeem/OneDrive/Desktop/flask
=> => transferring context: 818B                                         0.0s
=> CACHED [2/5] WORKDIR /usr/src/app                                       0.0s
=> [3/5] COPY package*.json ./                                              0.1s
=> [4/5] RUN npm install                                                 6.3s
=> [5/5] COPY . .                                                       0.1s
=> exporting to image                                                 1.7s
=> => exporting layers                                              0.8s
=> => exporting manifest sha256:9e3ca47834c1027f0f36606ee6016d2ac6dfc47ed8 0.0s
=> => exporting config sha256:b80c9d2675ad4beb7321a0c3e3b31959f4e92ae5604f 0.0s
=> => exporting attestation manifest sha256:65afc6a7d4c581773ec7caefca0a70 0.0s
=> => exporting manifest list sha256:95a3c2310764473421ace0d566263bb427b4 0.0s
=> => naming to docker.io/library/my-app2:latest                         0.0s
=> => unpacking to docker.io/library/my-app2:latest                      0.7s
```

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker run -d -p 8080:8080 --name app1 my-app1
2a817359abaaa7221e7e52f16dc6b5cd266af557b597846771b18b1a1c13d73d
```

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker run -d -p 8081:8080 --name app2 my-app2
a8873c9f91796f60a9653a41140cd56d71c4f92f9d7fb8a93a13f161095d40ee
```

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ |
```



5. You updated some code and want to restart your frontend, backend, and DB containers quickly. What is the easiest way to restart all services?

```
👉 docker-compose.yml X
C: > Users > azeem > OneDrive > Desktop > flask > 👉 docker-compose.yml
1
2   services:
3     frontend:
4       build: ./my-app1
5       ports:
6         - "8080:8080"
7
8     backend:
9       build: ./my-app2
10      ports:
11        - "8081:8080"
12
13   db:
14     image: mongo:6
15     container_name: my-mongo
16     ports:
17       - "27017:27017"
18     volumes:
19       - mongo-data:/data/db
20
21   volumes:
22     mongo-data:
23
```

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker compose restart
[+] Restarting 3/3
✓ Container flask-backend-1    St...
✓ Container my-mongo          Started
✓ Container flask-frontend-1   St...           2.1s
                                         1.7s
                                         2.3s
```

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker restart 4afa457dd5cb f85f723929d8
4afa457dd5cb
f85f723929d8
```

6. Your frontend has a new release, but you don't want to restart the backend or DB.
How can you update and rebuild only the frontend container?

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker compose up -d --build frontend
[+] Running 0/19s (2/3)                               docker:desktop-linux
[+] Building 4.2s (5/10)                             docker:desktop-linux
[+] Building 4.8s (12/12) FINISHED                   docker:desktop-linux
=> [frontend internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 320B                 0.1s
=> [frontend internal] load metadata for docker.io/library/nod 3.9s
=> [frontend auth] library/node:pull token for registry-1.dock 0.0s
=> [frontend internal] load .dockerignore           0.0s
=> => transferring context: 2B                     0.0s
=> [frontend 1/5] FROM docker.io/library/node:18@sha256:c6ae79 0.1s
=> => resolve docker.io/library/node:18@sha256:c6ae79e38498325 0.1s
=> [frontend internal] load build context          0.0s
=> => transferring context: 90B                   0.0s
=> CACHED [frontend 2/5] WORKDIR /usr/src/app       0.0s
=> CACHED [frontend 3/5] COPY package*.json ./      0.0s
=> CACHED [frontend 4/5] RUN npm install            0.0s
=> CACHED [frontend 5/5] COPY . .                  0.0s
=> [frontend] exporting to image                   0.2s
=> => exporting layers                          0.0s
=> => exporting manifest sha256:d5a0c552bd39e2c4b9827ce8313269 0.0s
=> => exporting config sha256:8b2d40b0024d3fd5dc76da3134967387 0.0s
=> => exporting attestation manifest sha256:b22fe29976cbb0fb19 0.1s
=> => exporting manifest list sha256:20dd3b8f864e75f6ed1d5bc6f 0.0s
=> => naming to docker.io/library/flask-frontend:latest 0.0s
[+] Running 2/2g to docker.io/library/flask-frontend:latest 0.0s
  ✓ Service frontend      Built 6.1s
  ✓ Container flask-frontend-1 S... 2.3s
```

```
azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$
```

7. Tomcat is listening on port 8080, but you get a "Connection Refused" error in the browser. Why?



This site can't be reached

localhost refused to connect.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR_CONNECTION_REFUSED

[Reload](#)

[Details](#)

"Connection Refused" happens because **the server process (like Tomcat) is not running or not listening on the port (8080), or the port is blocked/not exposed.**

The screenshot shows a web browser window with the URL `localhost:8080` in the address bar. A message at the top says "If you're seeing this, you've successfully installed Tomcat. Congratulations!" Below this, there's a cartoon cat icon and some recommended reading links: "Security Considerations How-To", "Manager Application How-To", and "Clustering/Session Replication How-To". To the right, there are links for "Server Status", "Manager App", and "Host Manager". At the bottom, there are sections for "Developer Quick Start" (with links to "Tomcat Setup" and "First Web Application"), "Documentation" (with links to "Tomcat 9.0 Documentation", "Tomcat 9.0 Configuration", and "Tomcat Wiki"), "Examples" (with links to "Realms & AAA" and "JDBC DataSources"), "Servlet Specifications" (with a link to "Tomcat Versions"), and "Getting Help" (with a link to "FAQ and Mailing Lists"). The Apache logo is visible in the top right corner of the page content area.

8. You're not sure which container is using port 3000. How do you check?

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
AMES
967ff23dc7fe flask-frontend "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:8080->8080/tcp f
flask-frontend-1
4afa457dd5cb flask-backend "docker-entrypoint.s..." 11 minutes ago Up 4 minutes 0.0.0.0:8081->8080/tcp f
flask-backend-1
64d71b9a03a4 mongo:6 "docker-entrypoint.s..." 11 minutes ago Up 8 minutes 0.0.0.0:27017->27017/tcp m
y-mongo
6058c9611591 64825fb4d70a "docker-entrypoint.s..." 4 days ago Up 12 minutes 6379/tcp m
yapi3
8e0059dd6e07 64825fb4d70a "docker-entrypoint.s..." 4 days ago Up 12 minutes 6379/tcp m
yapi2
37ade3a69024 64825fb4d70a "docker-entrypoint.s..." 4 days ago Up 12 minutes 6379/tcp m
yapi
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$
```

Here tcp image is running on port 3000

9. How do you stop and clean up your running Tomcat container and image?

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker stop 37ade3a69024
37ade3a69024

azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker rm 37ade3a69024
37ade3a69024
```

10. You want to share your app with a teammate. What do you do with the Docker image?

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker tag my-app2:latest azeemafirdous/my-app2:latest
```

```
azeem@LAPTOP-MUVD4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker login
Authenticating with existing credentials...
Login Succeeded
```

```

azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker push azeemafirdous/my-app2:latest
The push refers to repository [docker.io/azeemafirdous/my-app2]
9f5f6f9cf6cf: Pushed
cda7f44f2bdd: Pushed
79b2f47ad444: Pushed
3e6b9d1a9511: Pushed
34f8cf879920: Pushed
37927ed901b1: Pushed
e23f099911d6: Pushed
c6b30c3f1696: Pushed
2b217bcd967d: Pushed
3697be50c98b: Pushed
e6f934343b41: Pushed
461077a72fb7: Pushed
232fa65e5ce5: Pushed
latest: digest: sha256:95a3c23107644734212ace0d566263bb427b4c2e871b83ef5302e389f9a0640d size: 856

```

Name	Last Pushed	Contains	Visibility	Scout
azeemafirdous/my-app2	5 minutes ago	IMAGE	Public	Inactive
azeemafirdous/redis1	4 days ago	IMAGE	Public	Inactive

So we are storing the image in docker hub whenever a team mate wants it they can pull the image

```

azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ docker pull azeemafirdous/redis1
Using default tag: latest
latest: Pulling from azeemafirdous/redis1
Digest: sha256:e39ec005579e3e7d42dc46e573716a2eb10e25e280de35160515d9a7fed4dd16
Status: Downloaded newer image for azeemafirdous/redis1:latest
docker.io/azeemafirdous/redis1:latest

azeem@LAPTOP-MUV4INQ MINGW64 ~/OneDrive/Desktop/flask (main)
$ 

```

Prepared by:

Azeema firdous

23BD1A05A4

CSE-F