INFN BARI
Istituto Nazionale di Fisica Nucleare
Sezione di Bari

Docker compose e architettura multi container

Basic concepts

Gioacchino Vino (INFN Bari)
gioacchino.vino@ba.infn.it

Outline

- Multi-Container Application
- Docker Compose
- References
- Hands-on

Multi-Container Application

Why Multi-container applications?

- Development environments
- Automated testing environments

Multi-Container Application: Motivation

What if your application requires multiple containers executed at once?

Surely, multiple “docker run“ commands can be used:

```
docker container run -d -p 3000:80 -v host_path1:container_path1 -e ENV_VAR=VAL1 container_image1
docker container run -d -p 3001:80 -v host_path2:container_path2 -e ENV_VAR=VAL2 container_image2
docker container run -d -p 3002:80 -v host_path3:container_path3 -e ENV_VAR=VAL3 container_image3
docker container run -d -p 3003:80 -v host_path4:container_path4 -e ENV_VAR=VAL4 container_image4
```

A script could be even written to implement “start”, “restart”, “stop” commands.

What if we update only a single container and we want to restart only that one?

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker application

- YAML file to configure your app's services

```
docker network create db_net
docker volume create db_data
docker container run -d \
–name influxdb  \
--network db_net \
  -v db_data:/var/lib/influxdb2 \
  -p 8086:8086
  influxdb:2.1
```

```
version: "3.8"
services:
  db:
    image: influxdb:2.1
    volumes:
      - db_data:/var/lib/influxdb2:rw
    ports:
      - "8086:8086"
    networks:
      - db_net

volumes:
  db_data:

networks:
  db_net:
```

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker application

- YAML file to configure your app's services
- With a single command, you create and start all the services of your application

```
docker compose up         #initialize and start the application environment
docker compose start      #start the app environment
docker compose restart    #restart the created app environment
docker compose stop       #stop the app environment
docker compose down       #stop and destroy the app environment
```

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker application

```
user@vm:~/prova_1$ cat docker-compose.yaml
version: "3.8"
services:
  db:
    image: influxdb:2.1
    volumes:
    - db_data:/var/lib/influxdb2.1:rw
    ports:
    - "8086:8086"
    networks:
    - db_net

volumes:
  db_data:

networks:
  db_net:
```

```
user@vm:~/prova_1$ docker compose up -d
✔ db 10 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]        0B/0B   Pulled 39.4s
[+] Running 3/3
✔ Network prova_1_db_net      Created
✔ Volume "prova_1_db_data"    Created
✔ Container prova_1-db-1      Started

user@vm:~/prova_1$ docker container ps
CONTAINERID   IMAGE            NAMES
Be72e5..      influxdb:2.1     prova_1_db_1

user@vm:~/prova_1$ docker volume ls
DRIVER  VOLUME NAME
local   46f4a..
local   prova_1_db_data

user@vm:~/prova_1$ docker network ls
NETWORK ID     NAME             DRIVER       SCOPE
51039c996454   bridge           bridge       local
85ffa41af96f   host             host         local
d6f329122060   none             null         local
ec31eefd60d5   prova_1_db_net   bridge       local
```

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker application

```
user@vm:~/prova_1$ cat docker-compose.yaml
version: "3.8"
services:
  db:
    image: influxdb:2.1
    volumes:
    - db_data:/var/lib/influxdb2.1:rw
    ports:
    - "8086:8086"
    networks:
    - db_net

volumes:
  db_data:

networks:
  db_net:
```

```
user@vm:~/prova_1$ docker compose down
[+] Running 2/2
 ✔ Container prova_1-db-1  Removed
 ✔ Network prova_1_db_net  Removed

user@vm:~/prova_1$ docker container ps
CONTAINERID   IMAGE            STATUS          NAMES

user@vm:~/prova_1$ docker volume ls
DRIVER  VOLUME NAME
local   46f4..
local   prova_1_db_data

user@vm:~/prova_1$ docker network ls
NETWORK ID     NAME    DRIVER  SCOPE
51039c996454   bridge  bridge  local
85ffa41af96f   host    host    local
d6f329122060   none    null    local
```

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker application

- Volumes are not removed with `docker compose down`
- Compose preserves all volumes used by your services
- Data inside volume is not lost and can be reused once the app is restarted
- To remove volume as well, use `docker compose down --volumes`

```
user@vm:~/prova_1$ docker compose down
[+] Running 2/2
 ✔ Container prova_1-db-1  Removed
 ✔ Network prova_1_db_net  Removed

user@vm:~/prova_1$ docker container ps
CONTAINERID   IMAGE            STATUS          NAMES

user@vm:~/prova_1$ docker volume ls
DRIVER  VOLUME NAME
local   46f4..
local   prova_1_db_data

user@vm:~/prova_1$ docker network ls
NETWORK ID     NAME    DRIVER  SCOPE
51039c996454   bridge  bridge  local
85ffa41af96f   host    host    local
d6f329122060   none    null    local
```

Docker Compose: Project name

Docker Compose is a tool for defining and running multi-container Docker application

- Compose uses a project name to isolate environments from each others
- If not set, the folder name is taken as project name
- A custom project name can be set using the **-p** command line option or the **COMPOSE_PROJECT_NAME** env var

```
user@vm:~$ cd ..
user@vm:~$ cp -r prova_1/ prova_2/
user@vm:~$ cd prova_2
user@vm:~/prova_2$ docker compose up -d
[+] Running 3/3
 ✔ Network prova_2_db_net     Created
 ✔ Volume "prova_2_db_data"   Created
 ✔ Container prova_2-db-1     Started

user@vm:~/prova_2$ docker ps
CONTAINERID   IMAGE             NAMES
297f44..      influxdb:2.1      prova_2_db_1

user@vm:~/prova_2$ docker compose down
[+] Running 2/2
 ✔ Container prova_2-db-1  Removed
 ✔ Network prova_2_db_net  Removed
```

Docker Compose: Network

- By default Compose sets up a single network for your app.
- Each container for a service joins the default network and is both **reachable** by other containers on that network and **discoverable** by them at a hostname identical to the container name.
- If you make a configuration change to a service and run `docker compose up` to update it, the old container is removed and the new one joins the network under a **different IP** address but the **same name**.
- Use **container names** and port to connect services
- You can specify your own networks with the top-level networks key.

```
networks:
  front:
    driver: bridge
    driver_opts:
      com.docker.network.enable_ipv6: "true"
    ipam:
      driver: default
      config:
        - subnet: 172.16.238.0/24
        gateway: 172.16.238.1
        - subnet: "2001:3984:3989::/64"
        gateway: "2001:3984:3989::1"
```

Docker Compose

Goal: Deploy Grafana and InfluxDB services using docker-compose starting from the following commands

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

Docker Compose

Services are components of our application and can be based on different image

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
```

Docker Compose

Network section

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    networks:
      - my_net
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    networks:
      - my_net

networks:
  my_net: {}
```

```
user@vm:~/prova_3$ docker network ls
NETWORK ID     NAME             DRIVER  SCOPE
52724293ae72   prova_3_my_net   bridge  local
```

Docker Compose

Network section

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
```

```
user@vm:~/prova_3$ docker network ls
NETWORK ID    NAME              DRIVER  SCOPE
0f4f49bfdccc  prova_3_default   bridge  local
```

Docker Compose

Volume section

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
      - influxdb-storage:/var/lib/influxdb
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana

volumes:
  influxdb-storage:
  grafana-storage:
```

Docker Compose

Environment variable section

With env vars defined inside docker-compose.yml

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
      - influxdb-storage:/var/lib/influxdb
    environment:
      - INFLUXDB_DB=db0
      - INFLUXDB_ADMIN_USER=admin
      - INFLUXDB_ADMIN_PASSWORD=admin
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=admin

volumes:
  influxdb-storage:
  grafana-storage:
```

Docker Compose

Environment variable section

With env vars defined outside docker-compose.yml

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          influxdb:2.1
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
      - influxdb-storage:/var/lib/influxdb
    environment:
      - INFLUXDB_DB=db0
      - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
      - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW}
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME}
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW}

volumes:
  influxdb-storage:
  grafana-storage:
```

Docker Compose

Environment variable section

With env vars defined inside .env file

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
cat .env
INFLUXDB_USERNAME=admin
INFLUXDB_PW=admin
GRAFANA_USERNAME=admin
GRAFANA_PW=admin
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
      - influxdb-storage:/var/lib/influxdb
    environment:
      - INFLUXDB_DB=db0
      - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
      - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW}
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana
    environment:
      - GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME}
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW}

volumes:
  influxdb-storage:
  grafana-storage:
```

Docker Compose

Environment variable section

With env vars defined inside .env file

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
```

```
docker volume create grafana-storage
export GRAFANA_USERNAME=admin
export GRAFANA_PW=admin
docker container run --name grafana
          -p 3000:3000 \
          -v grafana-storage:/var/lib/grafana \
          -e GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME} \
          -e GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PW} \
          grafana/grafana:8.3.4-ubuntu
```

```
user@vm:~/example$ cat influxdb.env
INFLUXDB_DB=db0
INFLUXDB_USERNAME=admin
INFLUXDB_PW=admin
user@vm:~/example$ cat grafana.env
GRAFANA_USERNAME=admin
GRAFANA_PW=admin
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
      - influxdb-storage:/var/lib/influxdb
    env_file:
      - influxdb.env
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    volumes:
      - grafana-storage:/var/lib/grafana
    env_file:
      - grafana.env

volumes:
  influxdb-storage:
  grafana-storage:
```

Docker Compose

Environment variable section

- Use env vars to customize your application

```
services:
  influxdb:
    image: influxdb:${INFLUXDB_VERSION}
```

- If an environment variable is not set, Compose substitutes with an empty string

```
services:
  influxdb:
    image: influxdb:${INFLUXDB_VERSION:-2.1}
```

- Mixed approaches are allowed. Compose uses the following priority order, overwriting the less important with the higher ones:
  1. Compose file
  2. Shell environment variables
  3. Environment file
  4. Dockerfile
  5. Variable not defined

Docker Compose

Health Checks

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
          -p 8086:8086 \
          -v influxdb-storage:/var/lib/influxdb \
          -e INFLUXDB_DB=db0 \
          -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
          -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
          --health-cmd='curl -f http://localhost:8086||exit 1' \
          --health-start-period=30s \
          --health-interval= 30s \
          --health-timeout=10s \
          --health-retries=4 \
          influxdb:2.1
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
       - influxdb-storage:/var/lib/influxdb
    environment:
       - INFLUXDB_DB=db0
       - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
       - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PASSWORD}
    healthcheck:
      test: "curl --fail http://localhost:8086 || exit 1"
      start_period: 30s
      interval: 30s
      timeout: 10s
      retries: 4
```

Docker Compose

Dependencies

Some applications need a dependency chain between services, so that some services get loaded before (and unloaded after) other ones.

```
cat .env
INFLUXDB_USERNAME=admin
INFLUXDB_PASSWORD=admin
GRAFANA_USERNAME=admin
GRAFANA_PASSWORD=admin
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
       - influxdb-storage:/var/lib/influxdb
    environment:
       - INFLUXDB_DB=db0
       - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
       - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PASSWORD}
  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ports:
      - '3000:3000'
    volumes:
       - grafana-storage:/var/lib/grafana
    environment:
       - GF_SECURITY_ADMIN_USER=${GRAFANA_USERNAME}
       - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PASSWORD}
    depends_on:
      - influxdb

volumes:
  influxdb-storage:
  grafana-storage:
```

Docker Compose

Waiting for a service is ready

Some applications might need to wait for a tool is “ready”, not just running.

Workarounds are available.

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
       - influxdb-storage:/var/lib/influxdb
    environment:
       - INFLUXDB_DB=db0
       - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
       - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PASSWORD}

  myapp:
    image: myapp-python:3.9
    depends_on:
       - influxdb
    command:
       - ["./wait-for-it.sh", "influxdb:8086", "--", "python",
"/app.py"]
```

Docker Compose

Waiting for a service is ready

Some applications might need to wait for a tool is “ready”, not just running.

Best solution:

Insert the condition to wait until the service becomes “HEALTHY”

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
       - influxdb-storage:/var/lib/influxdb
    environment:
       - INFLUXDB_DB=db0
       - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
       - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PASSWORD}
    healthcheck:
      test: "curl --fail http://localhost:8086 || exit 1"
      start_period: 30s
      interval: 30s
      timeout: 10s
      retries: 4

  grafana:
    image: grafana/grafana:8.3.4-ubuntu
    ...
    depends_on:
      influxdb:
        condition: service_healthy
```

Docker Compose

Profiles

Profiles allow adjusting the Compose application model for various usages and environments by selectively enabling services.

This is achieved by assigning each service to zero or more profiles.

If unassigned, the service is always started but if assigned, it is only started if the profile is activated.

```
version: "3.8"
services:
  frontend:
    image: frontend
    profiles: ["frontend"]

  phpmyadmin:
    image: phpmyadmin
    depends_on:
      - db
    profiles:
      - debug

  backend:
    image: backend

  db:
    image: mysql
```

```
docker compose --profile frontend --profile debug up
#or
COMPOSE_PROFILES=frontend,debug docker compose up
```

Docker Compose

Build Dockerfile and run

The base image of a container can be defined also by building images using a Dockerfile

```
user@vm:~/prova_4$ ls -l
Dockerfile
requirements
docker-compose.yml
my-script.py
```

```
user@vm:~/prova_4$ cat Dockerfile
FROM python:3

COPY requirements /requirements
RUN pip install -r requirements
COPY my-script.py /my-script.py

CMD [ "python3", "/my-script.py"]
```

```
user@vm:~/prova_4$ cat requirements
pandas
numpy
```

```
user@vm:~/prova_4$ cat my-script.py
import socket
import sys
from time import sleep
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = ('localhost', 1234)
print(f'starting up on {server_address[0]} port {server_address[1]}')
sock.bind(server_address)
sleep(60)
```

```
user@vm:~/prova_4$ cat docker-compose.yml
version: '3.8'
services:
  my_service:
    build: .
    ports:
      - '1234:1234'
    volumes:
      - dir_data:/data
```

```
user@vm:~/prova_4$ docker compose up -d --build
Creating network "prova_4_default" with the default driver
Building my service
Sending build context to Docker daemon   5.12kB
Step 1/4 : FROM python:3
 ---> de529ffbdb66
Step 2/4 : COPY requirements ./
 ---> 8f6840fe4dab
Step 3/4 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in 363ffa40779c
Collecting pandas
  Downloading pandas-1.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.7 MB)
Collecting numpy
  Downloading numpy-1.22.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.8 MB)
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting pytz>=2020.1
  Downloading pytz-2021.3-py2.py3-none-any.whl (503 kB)
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pytz, python-dateutil, numpy, pandas
Successfully installed numpy-1.22.2 pandas-1.4.0 python-dateutil-2.8.2 pytz-2021.3 six-1.16.0
Removing intermediate container 363ffa40779c
 ---> 79f0ed627dd6
Step 4/4 : CMD [ "python3", "/my-script.py" ]
 ---> Running in 4f45fc08e6df
Removing intermediate container 4f45fc08e6df
 ---> 4fd169b43c5f
Successfully built 4fd169b43c5f
Successfully tagged prova_4_my_service:latest
Creating prova_4_my_service_1 ... done
```

Multi-Container Application: Motivation

Using Compose is basically a three-step process:

- Define your app's environment with a **Dockerfile** so it can be reproduced anywhere
- Define the services that make up your app in **docker-compose.yml** so they can be run together in an isolated environment
- Run **docker compose up** and the Docker compose command starts and runs your entire app.

Multi-Container Application: Motivation

Docker Compose features:

- Multiple isolated environments on a single host
  - Development host
  - CI
  - shared host
- Preserve volume data when containers are created
- Only recreate containers that have changed
- Orchestrate multiple containers that work together
- Using environment variables when moving solutions between environments

Multi-Container Application: Motivation

Docker Compose benefits:

- Easy management of multi-container applications
- Reproducible builds
- Improved collaboration
- Simplified deployment

References

- Overview of Docker Compose
- YAML - Wikipedia
- Overview of docker compose CLI
- Use Docker Compose

Hands-on

1. Write the docker-compose.yml file for the following docker cli

```
docker network create wordpress_net
docker volume create db_data
docker volume create wp_data
docker container run --name db \
  --network wordpress_net \
  -v db_data:/var/lib/mysql \
  -e MYSQL_ROOT_PASSWORD=somewordpress \
  -e MYSQL_USER=wordpress-user \
  -e MYSQL_PASSWORD=wordpress-password \
  -e MYSQL_DATABASE=wordpress-db \
  --restart always \
  --health-cmd="mysqladmin ping --silent" \
  --health-interval=10s \
  --health-start-period=10s \
  --health-timeout=10s \
  --health-retries=60 \
  --restart always \
  -d \
  mariadb:10.6.4-focal
```

```
docker run --name wp \
  --network wordpress_net \
  -v wp_data:/var/www/html
  -p 8080:80 \
  -e WORDPRESS_DB_HOST=db \
  -e WORDPRESS_DB_USER=wordpress-user \
  -e WORDPRESS_DB_PASSWORD=wordpress-password \
  -e WORDPRESS_DB_NAME=wordpress-database \
  -d \
  wordpress

# insert the depends_on condition on db health
```

Hands-on

2. Write the docker-compose.yml file that builds the following Dockerfile and uses it

```
base-image: jupyter/minimal-notebook
python module to install: pandas, numpy
expose port: 8888
command: /opt/conda/bin/python3.11 /opt/conda/bin/jupyter-lab \
           --no-browser \
           --allow-root \
           --NotebookApp.token='' \
           --NotebookApp.password=''
```

Hands-on solutions

1. Write the docker-compose.yml file for the following docker cli

```
version: '3.8'
services:
  db:
    image: mariadb:10.6.4-focal
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - wp_net
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=somewordpress
      - MYSQL_DATABASE=wordpress-database
      - MYSQL_USER=wordpress-user
      - MYSQL_PASSWORD=wordpress-password
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "--silent"]
      interval: 10s
      timeout: 10s
      retries: 60
      start_period: 10s
```

```
  wordpress:
    image: wordpress:latest
    volumes:
      - wp_data:/var/www/html
    ports:
      - 8081:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=wordpress-user
      - WORDPRESS_DB_PASSWORD=wordpress-password
      - WORDPRESS_DB_NAME=wordpress-database
    depends_on:
      db:
        condition: service_healthy

volumes:
  db_data:
  wp_data:
```

Hands-on solutions

2. Write the docker-compose.yml file that builds the following Dockerfile and uses it

cat requirements.txt

```
pandas
numpy
```

cat Dockerfile

```
FROM jupyter/minimal-notebook
COPY requirements.txt /requirements.txt
RUN pip install -r /requirements.txt
```

cat docker-compose.yaml

```
version: '3.8'
services:
  my_jupyter:
      build: .
      ports:
        - 8888:8888
      command:
        - /opt/conda/bin/python3.11
        - /opt/conda/bin/jupyter-lab
        - --no-browser --allow-root
        - --NotebookApp.token=''
        - --NotebookApp.password=''
```