# Teamwork

Docker Compose

Romain Eggermont

WIZARDS TECHNOLOGIES
Sorciers du code

## Summary

## Introduction

A modern web app is often composed of multiple services: backend, frontend, bdd, Redis cache, ...

On a Docker environment, it can be tricky as manually manage every service to behave as expected

As an answer, we have tools called orchestrators that ease the environment configuration, local or production.

While Docker Swarm and Kubernetes are more focused for production, **docker-compose** is the orchestrator you want for local development

# Docker Compose - Config example

docker-compose.yaml

```yaml
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - “8000:8000”
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:${POSTGRES_PASSWORD}@postgres/${POSTGRES_DATABASE}"
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose - Build / Image

One of the two keys is mandatory:

- Build means the image used by the service is built from a Dockerfile (and automatically built by docker-compose if not already done) and accepts a path as a value.
- Image means the image used by the service is downloaded from a registry (default: Docker Hub)

```yaml
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - “8000:8000”
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:[...]”
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose - Environment

List of environment variables to be used by the service.

- The syntax is basic: “`<env_var_name>=<value>`”
- The value can be:
  - Plain text
  - An environment variable from your computer (syntax: `${ENV_VAR}`
  - Mixed value of above types (see DATABASE_URL)

```yaml
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - “8000:8000”
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:[...]”
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose - Environment

Docker Compose checks for environment variables in your computer according to this priority order:

- An .env file in the same folder as the docker-compose.yaml
- Environment variables in your machine

```yaml
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - "8000:8000"
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:[...]"
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose - Ports

List of binding ports between the service / container and the host machine

- The syntax is closed to environment parameter: “<host-port>:<container-port>”
- The value can be:
  - Plain text
  - An environment variable from your computer (syntax: ${ENV_VAR}

```yaml
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - “8000:8000”
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:[...]”
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose - Volumes

As you saw in the previous course, manually managing volumes can be tedious. Docker Compose allows us to easily use them by two different ways:

- Anonymously - Generally for sending data from your machine to your container. Syntax: `./<path>:/<path>`
- Named - Generally for saving data from the container to your machine. Name declared in the volumes section and used as references in the service configuration. Syntax: `<name>:/<path>`

```yaml
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - "8000:8000"
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:[...]"
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose - Honourable mentions

- `version`: minimal docker-compose version compatibility
- `services`: declare different services
- `working_dir`: define default directory to work in
- `symfony_app` / `postgres`: name of the service
- `container_name`: name of the future docker container
- `depends_on`: define a container start order (here, we want postgres to be up before starting symfony application)

```
version: '3'
services:
  symfony_app:
    container_name: symfony_app
    build: docker/apache
    working_dir: /var/www/html
    volumes:
      - ./:/var/www/html
    ports:
      - “8000:8000”
    environment:
      - "DATABASE_URL=postgres://${POSTGRES_USER}:[...]”
    depends_on:
      - postgres
  postgres:
    container_name: postgres
    image: postgres:13.1-alpine
    environment:
      - "POSTGRES_DATABASE=${POSTGRES_DATABASE}"
      - "POSTGRES_PASSWORD=${POSTGRES_PASSWORD}"
      - "POSTGRES_USER=${POSTGRES_USER}"
    volumes:
      - dbdata:/var/lib/postgresql/data
volumes:
  dbdata:
```

## Docker Compose commands

Docker Compose commands are not so far from Docker commands:

- `docker-compose build` - Build all Dockerfiles (if not done already)
- `docker-compose up` / `up <service-name>` - Run all services / Up a specific service (and its dependencies) (build Dockerfiles if not done already)
- `docker-compose down` / `down <service-name>` - Stop and kill all services / Stop and kill a specific service
- `docker-compose logs` - Structure and show logs for all services' containers
- `docker-compose exec <service_name> <command>` - Execute a specific command in a specific service's container (If container is running)
- `docker-compose run <service_name> <command>` - Execute a specific command in a specific service's container (Up the service in a container if not available)

## Final Project

Write a working (of course) docker-compose conf to collaborate in a specific project. Requirements:

- All requirements from Git Project (Git Flow, signed commits, issues, ...)
- Working docker-compose configuration (app + database)
- Working Todo CRUD
- Todo elements saved in a database

Expected delivery on MyGES: Only GitHub / GitLab link

Due date: Friday 22nd - 3:00PM