Deploy dự án 101 v2

By GiangDQ - SE172256

A. Setup vps:

1. Chuẩn bị hệ thống

1.1 Cập nhật hệ thống

```
apt update && sudo apt upgrade -y
```

1.2 Cài đặt các gói cần thiết

```
 apt install apt-transport-https ca-certificates curl softw
are-properties-common -y
```

2. Cài đặt Docker

2.1 Thêm GPG key và repository của Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | g
pg --dearmor -o /usr/share/keyrings/docker-archive-keyring.
gpg && echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg] https://
download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
| tee /etc/apt/sources.list.d/docker.list > /dev/null
```

> **Quan trọng:** Lệnh này thêm key GPG và repository chính thức của Docker. Đây là cách được khuyến nghị bởi Docker để đảm bảo tính xác thực và an toàn khi cài đặt.

2.2 Cài đặt Docker

```
apt update && apt install docker-ce docker-ce-cli container
d.io -y
```

> Nếu gặp lỗi này khi chạy:
>
> ```
> Package docker-ce is not available, but is refe
> This may mean that the package is missing, has
> is only available from another source
>
> E: Package 'docker-ce' has no installation cand
> E: Unable to locate package docker-ce-cli
> E: Unable to locate package containerd.io
> E: Couldn't find any package by glob 'container
> ```
>
> 1. Cập nhật chỉ mục gói apt và cài đặt các gói cần thiết:
>
>    ```
>    apt update && apt install ca-certificates
>    curl gnupg
>    ```
>
> 2. Thêm khóa GPG chính thức của Docker:
>
>    ```
>    install -m 0755 -d /etc/apt/keyrings
>    curl -fsSL <https://download.docker.com/l
>    inux/ubuntu/gpg> | gpg --dearmor -o /etc/
>    apt/keyrings/docker.gpg
>    chmod a+r /etc/apt/keyrings/docker.gpg
>    ```
>
> 3. Thiết lập kho lưu trữ Docker:

```
echo \\
  "deb [arch="$(dpkg --print-architectur
e)" signed-by=/etc/apt/keyrings/docker.gp
g] <https://download.docker.com/linux/ubu
ntu> \\
  "$(. /etc/os-release && echo "$VERSION_
CODENAME")" stable" | \\
  tee /etc/apt/sources.list.d/docker.list
> /dev/null
```

4. Cập nhật lại chỉ mục gói apt:

```
apt update
```

5. Cài đặt Docker:

```
apt install docker-ce docker-ce-cli conta
inerd.io docker-buildx-plugin docker-comp
ose-plugin
```

2.3 Kiểm tra cài đặt Docker

```
docker --version
```

> **Lưu ý:** Nếu lệnh trên hiển thị phiên bản Docker, cài đặt đã thành công.

3. Cài đặt Docker Compose

3.1 Tải và cài đặt Docker Compose

```
curl -L "https://github.com/docker/compose/releases/downloa
d/v2.23.3/docker-compose-$(uname -s)-$(uname -m)" -o /usr/l
ocal/bin/docker-compose
```

> **Quan trọng:**
>
> - Kiểm tra phiên bản mới nhất tại https://github.com/docker/compose/releases và thay đổi số phiên bản nếu cần.
> - Nếu lệnh cuối hiển thị phiên bản Docker Compose, cài đặt đã thành công.

3.2 Cấp quyền thực thi cho Docker Compose

```
chmod +x /usr/local/bin/docker-compose
```

3.3 Kiểm tra cài đặt Docker Compose

```
docker-compose --version
```

> Lưu ý: Nếu lệnh trên hiển thị phiên bản Docker Compose, cài đặt đã thành công.

B. Setup Postgres và Redis

1. Thiết lập PostgreSQL

Tải image PostgreSQL:

```
docker pull postgres:latest
```

Giải thích: Lệnh này tải phiên bản mới nhất của image PostgreSQL từ Docker Hub.

Chạy container PostgreSQL:

```
docker run --name postgres-container -e POSTGRES_PASSWORD=yourpassword -p 5432:5432 -v postgres-data:/var/lib/postgresql/data -d postgres:latest
```

Giải thích các flag:

- `-name postgres-container`: Đặt tên cho container
- `e POSTGRES_PASSWORD=yourpassword`: Thiết lập mật khẩu cho tài khoản postgres
- `p 5432:5432`: Ánh xạ cổng 5432 của máy host với cổng 5432 của container
- `d`: Chạy container trong chế độ detached (chạy nền)

2. Thiết lập Redis

Tải image Redis:

```
docker pull redis:latest
```

Giải thích: Lệnh này tải phiên bản mới nhất của image Redis từ Docker Hub.

Chạy container Redis:

```
docker run --name redis-container -p 6379:6379 -d redis:latest
```

Giải thích các flag:

- `-name redis-container`: Đặt tên cho container
- `p 6379:6379`: Ánh xạ cổng 6379 của máy host với cổng 6379 của container
- `d`: Chạy container trong chế độ detached (chạy nền)

💡 Sau khi làm setup xong ta sẽ có 2 thứ:

- PostgreSQL:
  - User: postgres
  - Password: yourpassword (mật khẩu bạn đã đặt trong lệnh docker run)
  - Host: IP của máy host
  - Port: 5432
  - Database: postgres (database mặc định)
- Redis:
  - Host: IP của máy host
  - Port: 6379

C. Build và Push project lên Docker

Cấu hình Spring Boot project

1. Tạo các file cấu hình YAML

Trong thư mục `src/main/resources`, tạo 3 file sau:

application.yml

```
spring:
  profiles:
    active: dev

server:
  port: 8080

# Các cấu hình chung khác
```

application-dev.yml

```
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/devdb
    username: devuser
    password: devpassword
```

application-prod.yml

```
spring:
  datasource:
    url: jdbc:postgresql://prodhost:5432/proddb
    username: produser
    password: prodpassword
```

2. Cấu hình IntelliJ IDEA để chạy với profile "dev"

1. Mở Run/Debug Configurations cho ứng dụng của bạn.
2. Tìm đến mục "Environment variables".
3. Thêm biến môi trường mới:
   - Name: `SPRING_PROFILES_ACTIVE`
   - Value: `dev`

3. Chạy ứng dụng

Khi chạy ứng dụng từ IntelliJ IDEA, nó sẽ sử dụng profile "dev".

Build và push Spring Boot project

1. Cập nhật Dockerfile

Tạo hoặc cập nhật file `Dockerfile` trong thư mục root của dự án:

```
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY build/libs/*.jar app.jar
ENTRYPOINT ["java","-jar","/app/app.jar"]
```

2. Build ứng dụng Spring Boot

```
./gradlew clean build -x test
```

- `clean`: Xóa thư mục build cũ
- `build`: Biên dịch, test và đóng gói ứng dụng
- `-x test` : Bỏ qua việc test khi build

3. Build Docker image

```
docker build -t yourusername/your-project-name:latest .
```

- `t`: Gắn tag cho image
- `.`: Sử dụng Dockerfile trong thư mục hiện tại

4. Đăng nhập vào Docker Hub

```
docker login
```

Yêu cầu nhập username và password Docker Hub

5. Push image lên Docker Hub

```
docker push yourusername/your-project-name:latest
```

Đẩy image đã được tag lên Docker Hub

Lưu ý:

- Thay `yourusername` bằng tên người dùng Docker Hub của bạn.
- Thay `your-project-name` bằng tên dự án của bạn.
- Đảm bảo đã tạo repository trên Docker Hub trước khi push.
- Có thể thêm tag version cụ thể nếu cần (ví dụ: `1.0.0`).

D. Run project on docker

1. Đăng nhập vào VPS với quyền root

2. Tạo thư mục cho ứng dụng và file docker-compose.yml:

```
mkdir /opt/myapp && cd /opt/myapp
nano docker-compose.yml
```

3. Thêm nội dung sau vào file docker-compose.yml:

```
version: '3'
services:
  nginx:
    image: nginx:latest
    container_name: nginx-proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /path/to/nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - /path/to/ssl:/etc/nginx/ssl:ro
    depends_on:
      - app
    restart: always

  app:
    image: yourusername/your-project-name:latest
    container_name: myapp
    expose:
      - "8080"
    environment:
      - SPRING_PROFILES_ACTIVE=prod
    restart: always

  watchtower:
    image: containrrr/watchtower
    volumes:
```

```
      - /var/run/docker.sock:/var/run/docker.sock
    command: --interval 30
    restart: always
```

Giải thích:

- `app`: Cấu hình cho ứng dụng Spring Boot.
  - `image`: Chỉ định Docker image cho ứng dụng.
  - `container_name`: Đặt tên cho container.
  - `expose`: Mở cổng 8080 để các service khác trong mạng Docker có thể truy cập.
  - `environment`: Thiết lập biến môi trường để chạy ứng dụng với profile "prod".
  - `restart`: Luôn khởi động lại container nếu nó dừng hoạt động.
- `watchtower`: Dịch vụ tự động cập nhật container.
  - `image`: Sử dụng image containrrr/watchtower.
  - `volumes`: Kết nối Docker socket để Watchtower có thể tương tác với Docker daemon.
  - `command`: Thiết lập khoảng thời gian kiểm tra cập nhật là 30 giây.
  - `restart`: Luôn khởi động lại container nếu nó dừng hoạt động.
- `nginx`: Cấu hình cho máy chủ web Nginx làm reverse proxy.
  - `image`: Sử dụng phiên bản mới nhất của Nginx.
  - `container_name`: Đặt tên cho container Nginx.
  - `ports`: Ánh xạ cổng 80 và 443 của host vào container.
  - `volumes`:
    - Gắn file cấu hình Nginx từ host vào container.
    - Gắn thư mục chứa chứng chỉ SSL từ host vào container.
  - `depends_on`: Đảm bảo rằng service `app` được khởi động trước Nginx.
  - `restart`: Luôn khởi động lại container nếu nó dừng hoạt động.

4. Tạo file config cho nginx

Sử dụng lệnh `nano nginx.conf` và copy đoạn text dưới sau vào:

```
events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        listen 443 ssl;
        server_name example.com;

        ssl_certificate /etc/nginx/ssl/example.crt;
        ssl_certificate_key /etc/nginx/ssl/example.key;

        location / {
            proxy_pass http://app:8080;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }
    }
}
```

5. Thêm cert và key để cấp SSL

6. Chạy ứng dụng với Docker Compose:

```
docker-compose up -d
```

Flag `-d` chạy các container trong chế độ detached (nền).

Giải thích về Watchtower:

Watchtower là một ứng dụng tự động cập nhật Docker container. Nó theo dõi các image được sử dụng bởi các container đang chạy và tự động pull phiên bản

mới nhất của image từ Docker Hub. Khi phát hiện image mới, Watchtower sẽ tự động dừng và khởi động lại container với image mới.

Trong cấu hình trên:

- `-interval 30`: Watchtower sẽ kiểm tra cập nhật mỗi 30 giây.
- Khối `volumes` cho phép Watchtower tương tác với Docker daemon.

7. Cập nhật project khi có thay đổi

1. Push image mới lên Docker Hub (như trong phần C).
2. Watchtower sẽ tự động phát hiện image mới trên Docker Hub và cập nhật container trên VPS.

Lưu ý:

- Thay `yourusername/your-project-name` bằng tên image Docker.
- Điều chỉnh cổng `8080:8080` nếu ứng dụng sử dụng cổng khác.
- Đảm bảo bảo mật khi sử dụng Watchtower trong môi trường sản xuất.