

IR-03_ProdDesc: TRIỂN KHAI ỨNG DỤNG TRÊN DOCKER SWARM

Nội dung hướng dẫn bao gồm:

- 1: Backend
- 2: Frontend
- 3: Triển khai và kiểm tra
- 4: Quản lý stack
- Lưu ý quan trọng

PHẦN 1: BACKEND

BƯỚC 1: CHUẨN BỊ CODE CRAWLER (TRÊN MÁY THẬT)

- Code folder crawl (scrapy) (tự thực hiện)
- Build và push image:

```
# Login vào Docker Hub
docker login

# build image
docker build -t phamthixuanhien94/unitop:latest .

# Push image lên Docker Hub
docker push phamthixuanhien94/unitop:latest

# Kiểm tra image đã build:
docker images | grep unitop
```

Vd: crawl data từ unitop

phamthixuanhien94 / Repositories / unitop / General Using 1 of 1 private repositories.

phamthixuanhien94/unitop

Last pushed 1 minute ago · Repository size: 413.2 MB

Add a description

Add a category

Docker commands [Public view](#)

To push a new tag to this repository:

```
docker push phamthixuanhien94/unitop:tagname
```

General Tags Builds Collaborators Webhooks Settings

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	a minute ago	2 minutes ago

[See all](#)

Automated builds

Manually pushing images to Docker Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#) .

[Upgrade](#)

BƯỚC 2: TẠO STACK FILE

1. Tạo thư mục và file cấu hình trên manager node (192.168.56.10)

```
# Tạo thư mục project
mkdir -p ~/crawler-stack
cd ~/crawler-stack

# Tạo file docker-compose.yml
nano docker-compose.yml
```

2. Nội dung docker-compose.yml

```
version: '3.8'

services:
  mongodb:
    image: mongo:latest
    deploy:
      replicas: 2
      placement:
        constraints:
          - node.role!=manager
    volumes:
      - mongodb_data:/data/db
    networks:
      - crawler_network
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=password

  redis:
    image: redis:latest
    deploy:
      replicas: 2
      placement:
        constraints:
          - node.role!=manager
    volumes:
      - redis_data:/data
    networks:
      - crawler_network
    command: redis-server --requirepass password

  crawler:
    image: phamthixuanhien94/unitop-crawler:latest
    deploy:
      replicas: 3
      restart_policy:
```

```
    condition: on-failure
    delay: 5s
    max_attempts: 3
  depends_on:
    - mongodb
    - redis
  networks:
    - crawler_network
  environment:
    - MONGO_URI=mongodb://admin:password@mongodb:27017/
    - REDIS_HOST=redis
    - REDIS_PORT=6379
    - REDIS_PASSWORD=password

webui:
  image: phamthixuanhien94/crawler-ui:latest
  deploy:
    replicas: 2
  ports:
    - "80:80"
  networks:
    - crawler_network
  depends_on:
    - mongodb

networks:
  crawler_network:
    driver: overlay
    attachable: true

volumes:
  mongodb_data:
  redis_data:
```

PHẦN 2: FRONTEND

- Demo này tôi code bằng React, các bạn có thể chọn ngôn ngữ phù hợp cho mình.
- Frontend tôi code sau đó push lên dockerHub

Bước 1. Tạo react project

```
npx create-react-app crawler-ui  
cd crawler-ui
```

Bước 2. Install dependencies

```
npm install axios @material-ui/core @material-ui/icons
```

Bước 3. Code frontend

a. File src/app.js

```
import React, { useState, useEffect } from 'react';  
import axios from 'axios';  
import {  
  Container,  
  Grid,  
  Card,  
  CardContent,  
  Typography,  
  Rating  
} from '@material-ui/core';  
  
function App() {  
  const [courses, setCourses] = useState([]);  
  
  useEffect(() => {  
    const fetchCourses = async () => {  
      try {  
        const response = await axios.get('/api/courses');  
        setCourses(response.data);  
      } catch (error) {  
        console.error('Error fetching courses:', error);  
      }  
    };  
    fetchCourses();  
  }, []);  
  
  return (  
    <Container maxWidth="lg" style={{ marginTop: '2rem' }}>  
      <Typography variant="h4" gutterBottom>  
        Danh sách khóa học  
      </Typography>  
      <Grid container spacing={3}>  
        {courses.map((course) => (  
          <Grid item xs={12} sm={6} md={4} key={course._id}>
```

```

        <Card>
          <CardContent>
            <Typography variant="h6" gutterBottom>
              {course.coursename}
            </Typography>
            <Typography color="textSecondary">
              Giảng viên: {course.lecturer}
            </Typography>
            <Rating value={course.rating} readOnly />
            <Typography variant="body2" style={{marginTop: '1rem'}}>
              {course.intro}
            </Typography>
            <Typography variant="h6" color="primary" style={{marginTop:
'1rem'}}>
              {course.newfee}
            </Typography>
            <Typography variant="body2" color="textSecondary">
              Số bài học: {course.lesson_num}
            </Typography>
          </CardContent>
        </Card>
      </Grid>
    ))}
  </Grid>
</Container>
);
}

export default App;

```

Bước 4: Cấu hình Nginx

Tạo file nginx.conf:

```

server {
    listen 80;
    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

    location /api {
        proxy_pass http://crawler:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

```

Bước 5: Dockerfile cho Frontend (crawler-ui/Dockerfile):

```
FROM node:16-alpine as build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Bước 6: Build and push frontend image

```
docker build -t $DOCKER_USERNAME/crawler-ui:latest .
docker push $DOCKER_USERNAME/crawler-ui:latest
```

PHẦN 3: Triển khai và kiểm tra

1. DEPLOY STACK

```
docker stack deploy -c docker-compose.yml crawler_stack
```

2. KIỂM TRA TRIỂN KHAI

```
# Kiểm tra services
docker service ls

# Kiểm tra từng service
docker service ps crawler_stack_mongodb
docker service ps crawler_stack_redis
docker service ps crawler_stack_crawler
```

Sau khi deploy truy cập tại: <http://<manager-ip>:80>

---Ý nghĩa của việc dùng Stack trong Docker Swarm ---

Stack trong docker swarm là một nhóm các services liên quan được triển khai và chạy cùng nhau.

- Stack “crawler_stack” bao gồm 3 services:
 - Mongodb service (để lưu dữ liệu)
 - Redis service (để cache)
 - Crawler service (để crawl dữ liệu)

3. KIỂM TRA LOGS VÀ CONNECT

```
# Xem logs MongoDB
docker service logs crawler_stack_mongodb

# Xem logs Redis
docker service logs crawler_stack_redis

# Xem logs Crawler
docker service logs crawler_stack_crawler
```

```
# Kết nối vào MongoDB
docker exec -it $(docker ps -q -f name=mongodb) mongosh

# Kết nối vào Redis
docker exec -it $(docker ps -q -f name=redis) redis-cli
```

PHẦN 4: Quản lý Stack

Các lệnh cơ bản

```
# Triển khai stack
docker stack deploy -c docker-compose.yml crawler_stack

# Xem danh sách stacks
docker stack ls

# Xem các services trong stack
docker stack services crawler_stack

# Xóa stack
docker stack rm crawler_stack
```

- Ưu điểm của việc sử dụng Stack
 - Quản lý nhiều services liên quan như 1 đơn vị
 - Dễ dàng scale (tăng/giảm các containers)
 - Tự động cân bằng tải giữa các nodes
 - Dễ dàng update và rollback

Lưu ý quan trọng

1. Đảm bảo các ports không bị conflict
2. Kiểm tra kết nối giữa các services
3. Theo dõi logs thường xuyên
4. Backup dữ liệu định kỳ
5. Cập nhật security patches