HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA ĐÀO TẠO SAU ĐẠI HỌC

TRIỂN KHAI DỰ ÁN MICROSERVICE
MÔN HỌC: HỆ THỐNG PHÂN TÁN

GIẢNG VIÊN	TS. Kim Ngọc Bách	
THÀNH VIÊN NHÓM	Vũ Đức Uyên	B25CHHT068
	Nguyễn Mạnh Kỳ	B25CHHT033
	Nguyễn Tiến Đạt	B25CHHT010

Hà Nội, 15 tháng 12 năm 2025

Hướng Dẫn Triển Khai Dự Án Microservice

Tài liệu này hướng dẫn cách triển khai hệ thống Microservice E-Commerce lên môi trường production. Dự án sử dụng kiến trúc microservice với Docker Compose, RabbitMQ cho load balancing, và API Gateway RabbitMQ làm entry point chính.

Yêu Cầu Server

- **Tối Thiểu:**
  - CPU: 2 cores
  - RAM: 4GB
  - Disk: 20GB
  - OS: Ubuntu 20.04+ hoặc Windows Server 2019+
- **Khuyến Nghị (cho production):**
  - CPU: 4 cores
  - RAM: 8GB+
  - Disk: 50GB+ (SSD)
  - Network: Public IP với domain name
- **Phần Mềm Cần Cài:**
  - Docker và Docker Compose (bắt buộc)
  - Nginx (cho reverse proxy và serve frontend)
  - Git (để clone repository)
  - PostgreSQL Client (để test kết nối database)
- **External Services Cần Có:**
  - PostgreSQL Server (đã setup sẵn tại `47.130.33.106:5432`)
  - MongoDB Atlas (hoặc MongoDB server khác)
  - RabbitMQ Server (đã setup sẵn tại `47.130.33.106:5672`)

Chuẩn Bị Môi Trường

1. Cài Đặt Docker trên Ubuntu

Update hệ thống:
sudo apt-get update

  - sudo apt-get upgrade -y

Cài đặt Docker:
curl -fsSL https://get.docker.com -o get-docker.sh

  - sudo sh get-docker.sh

Thêm user vào docker group:
sudo usermod -aG docker $USER

  - newgrp docker # Hoặc logout/login lại

Cài đặt Docker Compose:
sudo curl -L
"https://github.com/docker/compose/releases/download/v2.20.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

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

Kiểm tra:
docker --version

- docker-compose --version

2. Cài Đặt Nginx

sudo apt-get install nginx -y
sudo systemctl start nginx
sudo systemctl enable nginx
sudo systemctl status nginx

3. Clone Repository

cd /opt
sudo git clone <your-repo-url> microservice
cd microservice
sudo chown -R $USER:$USER /opt/microservice Hoặc upload code lên server bằng FileZilla, WinSCP, etc.

4. Chuẩn Bị Database

Đảm bảo các databases đã được tạo trên PostgreSQL server (47.130.33.106):
-- Kết nối đến PostgreSQL server
psql -h 47.130.33.106 -U postgres

-- Tạo databases
CREATE DATABASE userservice_db;
CREATE DATABASE productservice_db;
CREATE DATABASE orderservice_db;

-- Kiểm tra
Kiểm tra kết nối từ server:
psql -h 47.130.33.106 -U postgres -d userservice_db

5. Kiểm Tra Rabbit

Test kết nối RabbitMQ
telnet 47.130.33.106 5672
Hoặc dùng curl (nếu có management plugin)

curl -u guest:guest http://47.130.33.106:15672/api/overview

Triển Khai Backend Services

Cách 1: Sử Dụng Docker Compose (Khuyến nghị)

Cập Nhật Thông Tin Kết Nối

Mở file `docker-compose.yml` và cập nhật các thông tin sau:
 # Thay đổi các giá trị này:
-
ConnectionStrings__PostgreSQL=Host=47.130.33.106;Port=5432;Database=userservice_db;Username=postgres;Password=YOUR_PASSWORD
- MongoDb__ConnectionString=YOUR_MONGODB_CONNECTION_STRING
- RabbitMQ__HostName=47.130.33.106

1. - RabbitMQ__Password=YOUR_RABBITMQ_PASSWORD
   Lưu ý: Thay `YOUR_PASSWORD` bằng password thực tế của PostgreSQL, `YOUR_MONGODB_CONNECTION_STRING` bằng connection string MongoDB Atlas, và `YOUR_RABBITMQ_PASSWORD` nếu RabbitMQ không dùng `guest/guest`.
2. **Chuyển Sang Production Mode**
   Để chạy production, thay đổi `ASPNETCORE_ENVIRONMENT` từ `Development` sang `Production` trong `docker-compose.yml` hoặc tạo file `docker-compose.production.yml` (như trong hướng dẫn cũ) với cấu hình tương tự.

Build và Chạy Services
cd /opt/microservice

```
# Build images (lần đầu hoặc khi có thay đổi code)
docker-compose build

# Chạy tất cả services
docker-compose up -d

# Kiểm tra status
docker-compose ps

# Xem logs
docker-compose logs -f

# Xem logs một service cụ thể

docker-compose logs -f api-gateway-rabbitmq
```

Kiểm Tra Services

Test API Gateway (entry point chính)

```
curl http://localhost:5010/health
```

Test User Service instances
curl http://localhost:5001/health
curl http://localhost:5004/health

Test Product Service instances
curl http://localhost:5002/health
curl http://localhost:5006/health

Test Order Service instances
curl http://localhost:5003/health
curl http://localhost:5007/health

Test API qua Gateway
curl http://localhost:5010/api/users
curl http://localhost:5010/api/products

- **Kiến Trúc Load Balancing:**
  - Dự án sử dụng **RabbitMQ Load Balancing** tự động.
  - Mỗi service có **2 instances** chạy song song.
  - API Gateway RabbitMQ (`5010`) là entry point duy nhất.
  - **Port Mapping:**
    - API Gateway RabbitMQ: `5010` (PRIMARY)
    - User Service: `5001`, `5004`
    - Product Service: `5002`, `5006`
    - Order Service: `5003`, `5007`
    - Frontend: `4200`

Cách 2: Chạy Trực Tiếp (Không Docker) - Không Khuyến Nghị

Publish Services

Publish từng service

cd Microservice.Services.UserService
dotnet publish -c Release -o /opt/microservice/user-service

... Tương tự cho các services khác ...

cd ../Microservice.ApiGateway.RabbitMQ

1. dotnet publish -c Release -o /opt/microservice/api-gateway

Tạo systemd Service Files

Tạo file `/etc/systemd/system/user-service-1.service` (và các file tương tự cho instances khác):

[Unit]

Description=User Service Instance 1

After=network.target

[Service]
Type=simple
WorkingDirectory=/opt/microservice/user-service
ExecStart=/usr/bin/dotnet
/opt/microservice/user-service/Microservice.Services.UserService.dll
Restart=always
RestartSec=10
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=ASPNETCORE_URLS=http://localhost:5001
Environment=ConnectionStrings__PostgreSQL=Host=47.130.33.106;Port=5432;Dat
abase=userservice_db;Username=postgres;Password=YOUR_PASSWORD

[Install]
WantedBy=multi-user.target
Sau đó:
sudo systemctl daemon-reload
sudo systemctl enable user-service-1
sudo systemctl start user-service-1

sudo systemctl status user-service-1

Cấu Hình Nginx Reverse Proxy

Nginx sẽ đóng vai trò reverse proxy cho API Gateway và serve static files cho Frontend.

Tạo Nginx Config

Tạo file `/etc/nginx/sites-available/microservice`:

API Gateway - Entry point chính

```
server {
    listen 80;
    server_name api.yourdomain.com; # Thay bằng domain của bạn hoặc IP

    location / {
        proxy_pass http://localhost:5010;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;

        # Timeout settings
        proxy_connect_timeout 60s;
```

```
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Health check endpoint
    location /health {
        proxy_pass http://localhost:5010/health;
        access_log off;
    }
}

# Frontend - Serve Angular app
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com; # Thay bằng domain của
bạn

    root /opt/microservice/Frontend/dist/microservice-frontend/browser;
    index index.html;

    # Gzip compression
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

    location / {
        try_files $uri $uri/ /index.html;
    }

    # Cache static assets

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # API proxy (nếu frontend gọi API qua cùng domain)
    location /api {
        proxy_pass http://localhost:5010;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

}
```

Enable Site
Tạo symbolic link
sudo ln -s /etc/nginx/sites-available/microservice /etc/nginx/sites-enabled/

# Xóa default site (optional)
sudo rm /etc/nginx/sites-enabled/default

# Kiểm tra config
sudo nginx -t

Reload Nginx

sudo systemctl reload nginx

Kiểm Tra
Test Nginx config
sudo nginx -t

Test API Gateway qua Nginx
curl http://api.yourdomain.com/health

Test Frontend

curl http://yourdomain.com

Setup SSL/HTTPS
Sử dụng Let's Encrypt để có SSL miễn phí:

Cài Đặt Certbot
sudo apt-get update
sudo apt-get install certbot python3-certbot-nginx -y

Lấy SSL Certificate
Lấy certificate cho API domain
sudo certbot --nginx -d api.yourdomain.com

Lấy certificate cho Frontend domain

sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com
Certbot sẽ tự động tạo SSL certificate, cập nhật Nginx config để sử dụng HTTPS, và setup auto-renewal.

Auto Renewal
Kiểm tra chức năng renew tự động của Certbot:
Test renewal (dry-run)
sudo certbot renew --dry-run

Kiểm tra cron job

sudo systemctl status certbot.timer

Triển Khai Frontend

Cách 1: Sử Dụng Docker (Khuyến nghị)

Frontend đã được Dockerized với multi-stage build.
Build và Chạy
cd /opt/microservice

Build và chạy frontend container
docker-compose up -d frontend

Kiểm tra

curl http://localhost:4200

Cấu hình API URL
Cấu hình biến môi trường `API_URL` trong `docker-compose.yml` để Docker inject API URL vào file `Frontend/src/app/config/environment.ts`:
Trong docker-compose.yml
frontend:
environment:

- API_URL=http://103.82.26.211:5010/api # Thay bằng domain/IP thực tế
Lưu ý: Nếu dùng domain với HTTPS, đổi thành: `https://api.yourdomain.com/api`.

Cách 2: Build và Deploy Static Files

Nếu muốn serve frontend bằng Nginx trực tiếp:
Build Production
cd /opt/microservice/Frontend

Install dependencies
npm install

Build production

npm run build -- --configuration production

Cập nhật API URL
Cập nhật URL API trong file `Frontend/src/app/config/environment.ts` trước khi build:
export const environment = {

```
  apiGatewayUrl: 'https://api.yourdomain.com',  // Thay bằng domain thực tế
  apiGatewayApiUrl: 'https://api.yourdomain.com/api',  // ... các URL khác};
```

Sau đó build lại: `npm run build -- --configuration production`

Copy files lên Nginx

```
# Copy built files
sudo cp -r dist/microservice-frontend/browser/* /var/www/html/

# Hoặc cấu hình Nginx trỏ đến thư mục build (đã cấu hình trong phần Nginx ở trên)
```

Monitoring và Logging

Xem Logs Docker

```
# Logs tất cả services
docker-compose logs -f

# Logs một service cụ thể
docker-compose logs -f api-gateway-rabbitmq
docker-compose logs -f user-service-1

# Logs với timestamp
docker-compose logs -f --timestamps

# Logs 100 dòng cuối

docker-compose logs --tail=100 user-service-1
```

Monitoring với Docker Stats

Xem resource usage của tất cả containers

```
docker stats

# Xem một container cụ thể
docker stats microservice-api-gateway-rabbitmq

# Xem với format tùy chỉnh

docker stats --format "table {{.Container}}\t{{.CPUPerc}}\t{{.MemUsage}}"
```

Setup Log Rotation

Tạo file `/etc/docker/daemon.json`:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  }

}
```

Restart Docker: `sudo systemctl restart docker`

Health Check Script

Tạo script kiểm tra health của services

`/opt/microservice/scripts/health-check.sh`:

#!/bin/bash

```
# health-check.sh

services=(
    "http://localhost:5010/health:API Gateway"
    "http://localhost:5001/health:User Service 1"
    # ... các services khác ...
)

echo "=== Health Check - $(date) ==="

for service in "${services[@]}"; do
    IFS=':' read -r url name <<< "$service"
    if curl -f -s -o /dev/null -w "%{http_code}" "$url" | grep -q "200"; then
        echo " $name is UP"
    else
        echo "$name is DOWN"
        # Có thể gửi email hoặc notification ở đây
    fi
done

echo "================================="
Chạy định kỳ với cron:
chmod +x /opt/microservice/scripts/health-check.sh
crontab -e

# Thêm dòng: */5 * * * * /opt/microservice/scripts/health-check.sh >>
/var/log/health-check.log 2>&1
```

Monitoring MongoDB và RabbitMQ

Test MongoDB connection

```
mongosh "YOUR_MONGODB_CONNECTION_STRING"

# Test RabbitMQ (nếu có management plugin)

curl -u guest:guest http://47.130.33.106:15672/api/overview
```

Backup và Recovery

Backup PostgreSQL Databases

Tạo script backup `/opt/microservice/scripts/backup-databases.sh`:

```
#!/bin/bash
# backup-databases.sh
```

```
BACKUP_DIR="/opt/backups"
DATE=$(date +%Y%m%d_%H%M%S)
PGHOST="47.130.33.106"
PGUSER="postgres"
PGPASSWORD="YOUR_PASSWORD"  # Thay bằng password thực tế

export PGPASSWORD

mkdir -p $BACKUP_DIR
# ... lệnh pg_dump cho từng database ...
# ... lệnh compress và xóa file cũ ...

echo "Backup completed: backup_$DATE.tar.gz"
unset PGPASSWORD
Setup cron để backup hàng ngày:
chmod +x /opt/microservice/scripts/backup-databases.sh
crontab -e

# Thêm: 0 2 * * * /opt/microservice/scripts/backup-databases.sh >>
/var/log/backup.log 2>&1
```

Restore Database

Extract backup

```
cd /opt/backups
tar -xzf backup_20240101_020000.tar.gz

# Restore từng database
export PGPASSWORD=YOUR_PASSWORD
psql -h 47.130.33.106 -U postgres -d userservice_db <
userservice_db_20240101_020000.sql
# ... các lệnh restore khác ...

unset PGPASSWORD
```

Backup Docker Images

Save images

```
docker save microservice-user-service-1:latest | gzip > user-service-image.tar.gz
# ... các images khác ...

# Load images (khi restore)

docker load < user-service-image.tar.gz
```

Troubleshooting

Services Không Khởi Động

```
# Kiểm tra logs
docker-compose logs user-service-1
```

Kiểm tra container status

- docker ps -a

Lỗi Kết Nối Database

Test kết nối từ server

psql -h 47.130.33.106 -U postgres -d userservice_db

# Kiểm tra firewall
sudo ufw status

- sudo ufw allow 5432/tcp # Nếu cần

Nginx 502 Bad Gateway

Kiểm tra services đang chạy

docker ps

Kiểm tra Nginx error logs

- sudo tail -f /var/log/nginx/error.log

Port Đã Được Sử Dụng

Tìm process đang dùng port

sudo lsof -i :5010

- sudo netstat -tulpn | grep :5010

Out of Memory

Kiểm tra memory usage

free -h
docker stats

Giới hạn memory cho containers trong docker-compose.yml
services:
  user-service-1:
    deploy:
      resources:
        limits:

- memory: 512M

RabbitMQ Connection Issues

Test kết nối RabbitMQ

telnet 47.130.33.106 5672

Kiểm tra RabbitMQ đang chạy

- curl -u guest:guest http://47.130.33.106:15672/api/overview

Checklist Triển Khai

Trước khi go-live, đảm bảo:

- Tất cả services đang chạy và healthy (`docker-compose ps`)
- Database connections hoạt động (test từng service)
- RabbitMQ connection OK (test từ Order Service)
- MongoDB connection OK (check logs)
- API Gateway RabbitMQ đang chạy trên port 5010
- Nginx reverse proxy cấu hình đúng
- SSL certificate đã được cài đặt (nếu dùng domain)
- Frontend build và deploy thành công
- Frontend có thể gọi API qua API Gateway
- Health checks đang chạy
- Backup script đã setup
- Firewall rules đã cấu hình (chỉ mở ports cần thiết)
- Test tất cả API endpoints
- Test authentication (login/register)
- Test từ frontend (tạo user, product, order)
- Load balancing hoạt động (2 instances mỗi service)

Tips và Best Practices

Security

1. **Environment Variables:** Không hardcode passwords trong code, dùng environment variables.
2. **Firewall:** Chỉ mở ports cần thiết (`80`, `443`, `22`).
3. **Passwords:** Đổi default passwords cho PostgreSQL, RabbitMQ.
4. **JWT Secret:** Đổi JWT secret key trong `appsettings.json`.
5. **HTTPS:** Luôn dùng HTTPS cho production.

Performance

1. **Resource Limits:** Đặt limits cho Docker containers.
2. **Gzip Compression:** Enable trong Nginx.
3. **Cache Static Assets:** Configure trong Nginx.
4. **Database Indexing:** Đảm bảo indexes được tạo.
5. **Connection Pooling:** Cấu hình trong connection strings.

Monitoring

1. **Health Checks:** Implement và monitor health endpoints.
2. **Logging:** Centralize logs, setup rotation.
3. **Alerts:** Setup alerts cho service downtime.
4. **Metrics:** Track response times, error rates.

Backup

1. **Automated Backups:** Setup cron jobs.
2. **Test Restores:** Định kỳ test restore process.
3. **Offsite Backups:** Lưu backups ở nơi khác.
4. **Backup Retention:** Giữ backups theo policy.

Tài Liệu Tham Khảo

- Docker Documentation
- Docker Compose Documentation
- Nginx Documentation
- Let's Encrypt
- .NET Production Best Practices
- RabbitMQ Documentation
- PostgreSQL Documentation

Hỗ Trợ

Nếu gặp vấn đề, kiểm tra:

1. Logs của services: `docker-compose logs -f <service-name>`
2. Health endpoints: `curl http://localhost:<port>/health`
3. Network connectivity: `docker network inspect microservice-network`
4. Container status: `docker ps -a`