**HƯỚNG DẪN TRIỂN KHAI HỆ THỐNG QUẢN LÝ**

**SINH VIÊN VỚI DOCKER SWARM**

*Thành Phố Hồ Chí Minh*, ngày 01 tháng 10 năm 2025

***Mục lục***

[1. Giới Thiệu Dự Án 1](#_Toc210291736)

[1.1 Mô Tả Tổng Quan 1](#_Toc210291737)

[1.2 Phạm Vi Dự Án 1](#_Toc210291738)

[1.3 Lý Do Sử Dụng Docker Swarm 1](#_Toc210291739)

[2. Thông Tin Nhà Phát Triển 2](#_Toc210291740)

[3. Kiến Trúc Hệ Thống 2](#_Toc210291741)

[3.1 Sơ Đồ Kiến Trúc (Mô Tả Văn Bản) 2](#_Toc210291742)

[Sơ đồ minh họa: 3](#_Toc210291743)

[3.2 Các Thành Phần Chính 3](#_Toc210291744)

[3.3 Tính Năng Chi Tiết 3](#_Toc210291745)

[3.4 Trường Dữ Liệu Sinh Viên 4](#_Toc210291746)

[4. Hướng Dẫn Triển Khai 5](#_Toc210291747)

[4.1 Điều Kiện Tiên Quyết 5](#_Toc210291748)

[4.2 Bước 1: Chuẩn Bị Môi Trường 5](#_Toc210291749)

[4.3 Bước 2: Cấu Hình File .env 6](#_Toc210291750)

[4.4 Bước 3: Phát Triển Cục Bộ (Local Development) 7](#_Toc210291751)

[4.5 Bước 4: Build và Push Images 8](#_Toc210291752)

[4.6 Bước 5: Triển Khai Với Docker Swarm 9](#_Toc210291753)

[5. Cấu Trúc Dự Án 10](#_Toc210291754)

[6. URL Truy Cập Và Thông Tin Đăng Nhập 12](#_Toc210291755)

[6.1 Môi Trường Phát Triển (Local) 12](#_Toc210291756)

[6.2 Môi Trường Production (Swarm) 12](#_Toc210291757)

[6.3 Thông Tin Đăng Nhập Mặc Định 12](#_Toc210291758)

[7. Giám Sát Và Quản Lý 12](#_Toc210291759)

[7.1 Lệnh Docker Swarm Cơ Bản 12](#_Toc210291760)

[7.2 Lệnh Debug Nâng Cao 13](#_Toc210291761)

[Exec vào Container: 13](#_Toc210291762)

[Logs Real-Time: 13](#_Toc210291763)

[Inspect Service: 14](#_Toc210291764)

[Tài Nguyên: 14](#_Toc210291765)

[Network: 14](#_Toc210291766)

[8. Bảo Mật Và Khuyến Nghị Production 14](#_Toc210291767)

[8.1 Các Biện Pháp Bảo Mật Cơ Bản 14](#_Toc210291768)

[8.2 Triển Khai HTTPS Chi Tiết 15](#_Toc210291769)

[9. Quy Trình CI/CD Đề Xuất 15](#_Toc210291770)

[10. Mở Rộng Hệ Thống 17](#_Toc210291771)

[10.1 Scale Horizontal (Thêm Replicas) 17](#_Toc210291772)

[10.3 Thêm Nodes Vào Cluster 18](#_Toc210291773)

[11. Xử Lý Sự Cố (Troubleshooting) 18](#_Toc210291774)

[11.1 Các Vấn Đề Thường Gặp Và Giải Pháp 18](#_Toc210291775)

[11.2 Quy Trình Debug Tổng Quát 19](#_Toc210291776)

[12. Kiểm Tra Hệ Thống (Health Checks) 19](#_Toc210291777)

[12.1 Lệnh Kiểm Tra Cơ Bản 19](#_Toc210291778)

[12.2 Health Check Trong Stack.yml 20](#_Toc210291779)

# 1. Giới Thiệu Dự Án

## 1.1 Mô Tả Tổng Quan

Dự án **Hệ Thống Quản Lý Sinh Viên** là một ứng dụng web đơn giản nhưng mạnh mẽ, được thiết kế để quản lý thông tin sinh viên trong môi trường giáo dục. Hệ thống sử dụng công nghệ hiện đại để đảm bảo tính khả dụng cao, dễ mở rộng và bảo mật.

* **Mục tiêu chính:** Cung cấp giao diện thân thiện để quản trị viên thêm, sửa, xóa và xem danh sách sinh viên.
* **Công nghệ cốt lõi:**
  + Backend: Flask (Python) với API RESTful.
  + Frontend: HTML/CSS thuần túy (không sử dụng framework JS để giữ sự đơn giản).
  + Database: PostgreSQL (hỗ trợ ACID và mở rộng tốt).
  + Orchestration: Docker Swarm (với 2 replicas để đảm bảo high availability).
  + Load Balancing: Nginx.
* **Lợi ích:** Hệ thống có thể chạy trên môi trường local, cloud (AWS, GCP) hoặc on-premise, với khả năng chịu lỗi và tự động scale.

## 1.2 Phạm Vi Dự Án

* **Tính năng cốt lõi:** Quản lý CRUD (Create, Read, Update, Delete) cho dữ liệu sinh viên.
* **Giới hạn:** Không hỗ trợ đa người dùng (chỉ admin), chưa tích hợp email/SMS thông báo. Có thể mở rộng trong tương lai.
* **Môi trường mục tiêu:** Phát triển local và production với Docker Swarm cluster (ít nhất 1 manager node và 1 worker node).

## 1.3 Lý Do Sử Dụng Docker Swarm

Docker Swarm là công cụ orchestration nhẹ nhàng từ Docker, phù hợp cho các dự án nhỏ đến trung bình. So với Kubernetes, Swarm dễ cấu hình hơn và tích hợp trực tiếp với Docker Compose.

# 2. Thông Tin Nhà Phát Triển

| **Thông Tin** | **Chi Tiết** |
| --- | --- |
| **Tên** | Nguyễn Minh Phúc |
| **Nghề Nghiệp** | Sinh viên năm 4, chuyên ngành Khoa học Máy tính |
| **Chuyên Môn** | DevSecOps (tập trung vào CI/CD, containerization, và bảo mật ứng dụng) |
| **Liên Hệ** | GitHub: [github.com/csenguyenminhphuc](https://github.com/csenguyenminhphuc) |
| **Kinh Nghiệm** | Đã triển khai các dự án tương tự với Docker, Flask, và PostgreSQL; tham gia các khóa học DevOps trên Coursera và Udemy. |

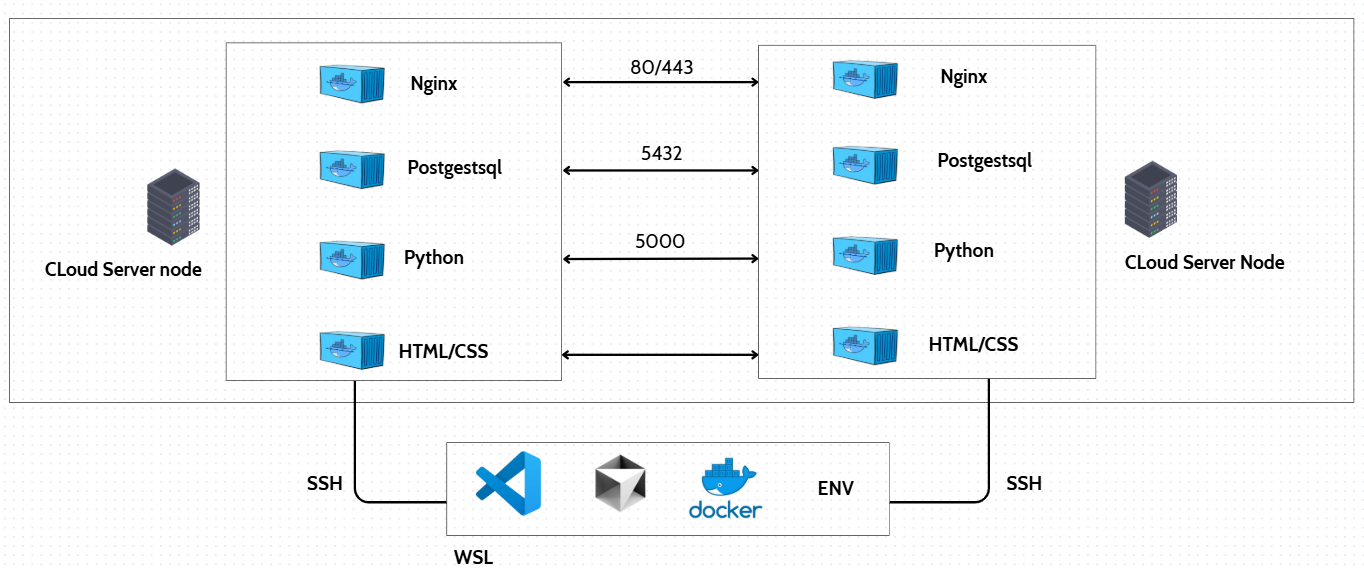
# 3. Kiến Trúc Hệ Thống

## 3.1 Sơ Đồ Kiến Trúc (Mô Tả Văn Bản)

Hệ thống được thiết kế theo mô hình microservices đơn giản, với Docker Swarm làm lớp orchestration:

* **Client** → **Nginx Load Balancer** (phân tải cho 2 replicas Flask) → **Flask App** (xử lý logic kinh doanh) → **PostgreSQL Database** (lưu trữ dữ liệu).
* **Mạng nội bộ:** Sử dụng overlay network của Swarm để các service giao tiếp an toàn.
* **Lưu trữ:** Images được lưu trên Docker Hub; dữ liệu DB persistent qua volumes.

## Sơ đồ minh họa:



## 3.2 Các Thành Phần Chính

| **Thành Phần** | **Mô Tả** | **Cấu Hình Mặc Định** |
| --- | --- | --- |
| **Frontend** | HTML/CSS thuần (templates Jinja2). | Static files trong /static/. |
| **Backend** | Flask API (routes: /login, /students). | Chạy trên port 5000, 4 workers (Gunicorn). |
| **Database** | PostgreSQL 13+. | DB: students\_db, User: postgres. |
| **Load Balancer** | Nginx proxy. | Cấu hình reverse proxy cho Flask replicas. |
| **Orchestration** | Docker Swarm. | Stack: student-management, 2 replicas cho webapp. |
| **Registry** | Docker Hub. | Images: csenguyenminhphuc/student-management-\*. |

## 3.3 Tính Năng Chi Tiết

1. **Trang Chủ:** Hiển thị giới thiệu cá nhân (tên, nghề nghiệp). Không yêu cầu auth.
2. **Xác Thực:** Form đăng nhập đơn giản (session-based). Sử dụng Flask-Login.
3. **Quản Lý Sinh Viên:**
   * **Xem Danh Sách:** Bảng hiển thị tất cả sinh viên (sortable theo ID).
   * **Thêm Mới:** Form với validation (tuổi > 16, GPA 0-4.0).
   * **Chỉnh Sửa:** Modal popup với pre-filled data.
   * **Xóa:** Xác nhận trước khi xóa (soft delete optional).

## 3.4 Trường Dữ Liệu Sinh Viên

| **Trường** | **Kiểu Dữ Liệu** | **Mô Tả** | **Ràng Buộc** |
| --- | --- | --- | --- |
| **MaSV** | VARCHAR(10) | ID duy nhất (e.g., SV001). | PRIMARY KEY, UNIQUE. |
| **HoTen** | VARCHAR(100) | Họ và tên đầy đủ. | NOT NULL. |
| **Tuoi** | INTEGER | Tuổi (18-25). | CHECK (tuoi > 0). |
| **NganhHoc** | VARCHAR(50) | Chuyên ngành (e.g., CNTT). | DEFAULT 'Khoa học Máy tính'. |
| **NamNhapHoc** | INTEGER | Năm nhập học (e.g., 2020). | CHECK (namnhaphoc >= 2000). |
| **GPA** | DECIMAL(3,2) | Điểm trung bình (0.00-4.00). | CHECK (gpa >= 0 AND gpa <= 4). |

**Ví dụ Dữ Liệu Mẫu (SQL Insert):**

sql

INSERT INTO students (masv, hoten, tuoi, nganhhoc, namnhaphoc, gpa)

VALUES ('SV001', 'Nguyễn Văn A', 20, 'CNTT', 2022, 3.5);

# 4. Hướng Dẫn Triển Khai

## 4.1 Điều Kiện Tiên Quyết

Trước khi bắt đầu, đảm bảo máy chủ đáp ứng:

* **Hệ Điều Hành:** Ubuntu 20.04+ hoặc Linux tương đương (không hỗ trợ Windows native).
* **Phần Mềm:**
  + Docker Engine: 20.10+ (cài đặt: curl -fsSL https://get.docker.com | sh).
  + Docker Compose: 2.0+ (sudo apt install docker-compose-plugin).
  + Git: Để clone repo.
* **Tài Khoản:** Docker Hub (tạo token access tại Settings > Security).
* **Tài Nguyên:** Ít nhất 4GB RAM, 2 CPU cores cho local testing.
* **Cảnh Báo:** Chạy với quyền sudo nếu cần; thêm user vào group docker để tránh sudo.

## 4.2 Bước 1: Chuẩn Bị Môi Trường

1. Mở terminal và clone repo:

bash

git clone https://github.com/csenguyenminhphuc/vti-final-project.git

cd vti-final-project

* + **Giải Thích:** Repo chứa tất cả source code, Dockerfiles và configs.
  + **Lưu Ý:** Nếu không có Git, tải ZIP từ GitHub và giải nén.

1. Tạo file .env từ mẫu:

bash

cp .env.example .env

* + **Giải Thích:** File này chứa biến môi trường để tránh hardcode secrets.

## 4.3 Bước 2: Cấu Hình File .env

Mở file .env bằng editor (e.g., nano .env) và chỉnh sửa. Dưới đây là mẫu chi tiết với giải thích:

env

# ===== CẤU HÌNH CƠ SỞ DỮ LIỆU =====

DB\_HOST=database # Tên service DB trong Swarm network

DB\_NAME=students\_db # Tên database

DB\_USER=postgres # User mặc định

DB\_PASSWORD=MatKhauBaoMat123! # MẬT KHẨU MẠNH: Kết hợp chữ, số, ký tự đặc biệt

DB\_PORT=5432 # Port chuẩn PostgreSQL

# ===== CẤU HÌNH FLASK =====

SECRET\_KEY=khoa-bi-mat-cuc-ky-bao-mat-thay-doi-trong-production-2024 # Tạo random: python -c "import secrets; print(secrets.token\_hex(16))"

FLASK\_ENV=production # production/development

# ===== THÔNG TIN ADMIN =====

ADMIN\_USERNAME=admin # Tên đăng nhập admin

ADMIN\_PASSWORD=admin123 # MẬT KHẨU MẠNH TRONG PROD

# ===== DOCKER HUB =====

DOCKER\_HUB\_USERNAME=csenguyenminhphuc # Username của bạn

DOCKER\_HUB\_TOKEN=your\_docker\_hub\_token\_here # Token từ Docker Hub (không dùng password)

IMAGE\_NAME=student-management-system

# ===== STACK SWARM =====

STACK\_NAME=student-management

REPLICAS=2 # Số replicas cho webapp (scale horizontal)

PORT=5000 # Port nội bộ Flask

WORKERS=4 # Số worker processes trong Gunicorn

# ===== CLOUDFLARE (TÙY CHỌN - CHO DNS/SSL) =====

API\_TOKEN=your\_cloudflare\_api\_token # Token từ Cloudflare API

ZONE\_ID=your\_cloudflare\_zone\_id # ID domain

RECORD\_ID=your\_cloudflare\_record\_id # ID DNS record

SUBDOMAIN=your-subdomain.your-domain.com # e.g., app.example.com

PRIMARY\_IP=your\_primary\_server\_ip # IP server chính

BACKUP\_IP=your\_backup\_server\_ip # IP backup

* **Lưu Ý Bảo Mật:**
  + Không commit .env (thêm vào .gitignore).
  + Sử dụng công cụ như pass hoặc Vault cho production.
  + Test config: echo $DB\_PASSWORD (nên ẩn output).

## 4.4 Bước 3: Phát Triển Cục Bộ (Local Development)

Sử dụng Docker Compose để test nhanh mà không cần Swarm.

1. Khởi động:

bash

docker-compose up -d

* + **Giải Thích:** -d chạy background. Compose sẽ build/pull images và tạo network.

1. Kiểm tra:

bash

docker-compose ps *# Xem containers đang chạy*

docker-compose logs -f *# Theo dõi logs real-time (Ctrl+C để dừng)*

1. Truy cập: Mở browser tại http://localhost (Nginx proxy).
   * **Test Tính Năng:** Đăng nhập admin → Thêm sinh viên mẫu → Kiểm tra DB qua pgAdmin hoặc psql.
2. Dừng:

bash

docker-compose down --volumes *# Xóa volumes nếu muốn reset DB*

* **Debug Local:** Nếu lỗi kết nối DB, kiểm tra docker network inspect <network\_name>.

## 4.5 Bước 4: Build và Push Images

Images cần được build trước khi deploy Swarm (để pull từ registry).

1. Set variables (thêm vào script bash nếu cần):

bash

export DOCKER\_HUB\_USERNAME=csenguyenminhphuc

export DOCKER\_HUB\_TOKEN=your\_token

1. Build từng image:

bash

*# App Flask*

docker build -t $DOCKER\_HUB\_USERNAME/student-management-app:latest .

*# Database (custom init scripts)*

docker build -f Dockerfile.db -t $DOCKER\_HUB\_USERNAME/student-management-db:latest .

*# Nginx*

docker build -f Dockerfile.nginx -t $DOCKER\_HUB\_USERNAME/student-management-nginx:latest .

* + **Giải Thích:** latest tag cho simplicity; dùng version-specific (e.g., v1.0) cho prod.

1. Login và Push:

bash

docker login -u $DOCKER\_HUB\_USERNAME -p $DOCKER\_HUB\_TOKEN

docker push $DOCKER\_HUB\_USERNAME/student-management-app:latest

*# Lặp lại cho db và nginx*

* + **Lưu Ý:** Push có thể mất 5-10 phút tùy kích thước image. Kiểm tra tại hub.docker.com.

## 4.6 Bước 5: Triển Khai Với Docker Swarm

Swarm cần ít nhất 1 node; khuyến nghị 3 nodes cho prod.

1. Khởi tạo Swarm (trên manager node):

bash

docker swarm init --advertise-addr <your-ip> *# e.g., 192.168.1.100*

* + **Giải Thích:** Lệnh này tạo token để join worker nodes.

1. Join Worker Nodes (nếu có cluster):

bash

docker swarm join --token <token-from-init> <manager-ip>:2377

1. Deploy Stack:

bash

docker stack deploy -c docker-stack.yml student-management

* + **Giải Thích:** File docker-stack.yml định nghĩa services, networks, volumes. Swarm sẽ tự pull images và schedule replicas.

1. Kiểm Tra:

bash

docker service ls *# Danh sách services*

docker stack ps student-management *# Tasks/replicas status*

docker node ls *# Nodes trong cluster*

* + **Thời Gian:** 1-2 phút để replicas healthy.
* **Truy Cập:** http://<swarm-ip> (Nginx expose port 80).

# 5. Cấu Trúc Dự Án

Dưới đây là cây thư mục chi tiết (có thể copy vào Word dưới dạng cây outline):

text

vti-final-project/

├── app.py # Flask app chính: Routes, models, auth logic

├── python.py # Helper scripts (e.g., data migration)

├── requirements.txt # Pip dependencies: flask, flask-login, psycopg2, gunicorn

├── templates/ # Jinja2 templates

│ ├── base.html # Layout chung (header, footer, CSS link)

│ ├── home.html # Trang giới thiệu (static content)

│ ├── login.html # Form login với error handling

│ └── dashboard.html # Bảng sinh viên + forms CRUD

├── static/ # Assets

│ └── css/

│ └── style.css # Responsive design (Bootstrap optional)

├── Dockerfile # Build Flask: FROM python:3.9, COPY ., pip install

├── Dockerfile.db # Custom Postgres: FROM postgres:13, COPY init-db.sql /docker-entrypoint-initdb.d/

├── Dockerfile.nginx # Nginx: FROM nginx:alpine, COPY nginx.conf /etc/nginx/

├── docker-compose.yml # Dev config: services với ports local

├── docker-stack.yml # Swarm config: deploy mode replicated, networks overlay

├── docker-stack-local.yml # Swarm local testing (single node)

├── docker-stack-simple.yml # Phiên bản đơn giản (1 replica)

├── docker-stack-fixed.yml # Fixed bugs (e.g., volume mounts)

├── nginx.conf # Proxy config: upstream flask { server webapp:5000; }, server block port 80

├── init-db.sql # CREATE TABLE students (...); INSERT admin user

├── .env # Secrets (gitignore)

├── .env.example # Mẫu env

├── .gitignore # Ignore .env, \_\_pycache\_\_, volumes

├── huongdan.txt # Hướng dẫn tiếng Việt ngắn gọn

└── README.md # Tài liệu này

**Ghi Chú:** Tổng kích thước repo ~50MB. Sử dụng tree command để generate cây thư mục.

# 6. URL Truy Cập Và Thông Tin Đăng Nhập

## 6.1 Môi Trường Phát Triển (Local)

| **URL** | **Mô Tả** | **Ghi Chú** |
| --- | --- | --- |
| <http://localhost> | Ứng dụng chính (qua Nginx) | Default route: Trang chủ. |
| <http://localhost:5000> | Truy cập trực tiếp Flask | Bỏ qua Nginx (cho debug). |
| localhost:5432 | PostgreSQL | Sử dụng pgAdmin hoặc DBeaver để connect. |

## 6.2 Môi Trường Production (Swarm)

| **URL** | **Mô Tả** | **Ghi Chú** |
| --- | --- | --- |
| http://<swarm-manager-ip> | Ứng dụng chính | Load balanced qua Nginx. |
| http://<swarm-ip>:8080 | Swarm Visualizer | Nếu enable trong stack.yml (optional). |

## 6.3 Thông Tin Đăng Nhập Mặc Định

* **Username:** admin
* **Password:** admin123

**Cảnh Báo:** Thay đổi ngay trong prod. Sử dụng bcrypt hash trong DB cho auth.

# 7. Giám Sát Và Quản Lý

## 7.1 Lệnh Docker Swarm Cơ Bản

| **Lệnh** | **Mô Tả** | **Ví Dụ Output** |
| --- | --- | --- |
| docker service ls | Xem services | ID, NAME, MODE, REPLICAS, IMAGE |
| docker stack ps student-management | Chi tiết tasks | TASK ID, NODE, CURRENT STATE (e.g., Running) |
| docker service logs student-management\_webapp | Logs service | ERROR/WARNING messages |
| docker service scale student-management\_webapp=4 | Scale replicas | Updated to 4/4 replicas |
| docker service update student-management\_webapp | Rolling update | Zero-downtime deploy |
| docker node ls | Nodes cluster | ROLE (Manager/Worker), STATUS |
| docker stack rm student-management | Remove stack | Graceful shutdown |

## 7.2 Lệnh Debug Nâng Cao

### Exec vào Container:

bash

CONTAINER\_ID=$(docker ps -q -f name=webapp)

docker exec -it $CONTAINER\_ID bash

*# Bên trong: python app.py --debug hoặc psql -U postgres*

### Logs Real-Time:

bash

docker service logs -f student-management\_webapp --since 10m *# Logs 10 phút qua*

### Inspect Service:

bash

docker service inspect student-management\_webapp --pretty

*# Kiểm tra ports, env vars, health checks*

### Tài Nguyên:

bash

docker stats --no-stream *# CPU/Mem usage*

docker system df *# Disk usage by images/volumes*

### Network:

bash

docker network ls *# Tìm overlay network*

docker network inspect student-management\_default *# Endpoints*

**Khuyến Nghị:** Tích hợp ELK Stack (Elasticsearch, Logstash, Kibana) cho logs production.

# 8. Bảo Mật Và Khuyến Nghị Production

## 8.1 Các Biện Pháp Bảo Mật Cơ Bản

1. **Mật Khẩu:** Luôn thay đổi defaults; sử dụng password manager.
2. **HTTPS:** Bắt buộc cho prod.
3. **Tường Lửa:** UFW trên Ubuntu: ufw allow 80,443,2377/tcp.
4. **Secrets:** Sử dụng Docker Secrets: echo "secret" | docker secret create db\_pass -.
5. **Backup:** Cron job: pg\_dump students\_db > backup.sql.
6. **Monitoring:** Logs với rotation; alert nếu replicas < 2.
7. **Updates:** docker image prune định kỳ; scan vulnerabilities với Trivy.

## 8.2 Triển Khai HTTPS Chi Tiết

1. Tạo self-signed cert (dev only):

bash

openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout nginx-selfsigned.key -out nginx-selfsigned.crt \

-subj "/C=VN/ST=HCM/L=HoChiMinh/O=Org/CN=localhost"

1. Cập nhật nginx.conf:

text

server {

listen 443 ssl;

ssl\_certificate /etc/nginx/nginx-selfsigned.crt;

ssl\_certificate\_key /etc/nginx/nginx-selfsigned.key;

...

}

1. Mount certs vào volume trong docker-stack.yml:

yaml

volumes:

- ./certs:/etc/nginx/certs:ro

* **Prod:** Sử dụng Let's Encrypt (Certbot) hoặc Cloudflare Origin CA.

# 9. Quy Trình CI/CD Đề Xuất

Sử dụng GitHub Actions cho automation. Dưới đây là workflow YAML đầy đủ (copy vào .github/workflows/deploy.yml):

yaml

name: Deploy to Production

on:

push:

branches: [main]

jobs:

build-and-deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Set up Docker Buildx

uses: docker/setup-buildx-action@v3

- name: Login to Docker Hub

uses: docker/login-action@v3

with:

username: ${{ secrets.DOCKER\_USERNAME }}

password: ${{ secrets.DOCKER\_HUB\_TOKEN }}

- name: Build and push images

run: |

docker buildx build --platform linux/amd64 -t ${{ secrets.DOCKER\_USERNAME }}/student-management-app:${{ github.sha }} . --push

# Lặp cho db và nginx

- name: Deploy to Swarm (via SSH)

uses: appleboy/ssh-action@v1.0.0

with:

host: ${{ secrets.SERVER\_IP }}

username: ${{ secrets.SERVER\_USER }}

key: ${{ secrets.SSH\_KEY }}

script: |

docker stack deploy -c docker-stack.yml student-management --with-registry-auth

* **Secrets:** Thêm DOCKER\_USERNAME, DOCKER\_HUB\_TOKEN, SERVER\_IP, etc. vào GitHub Repo Settings.
* **Lợi Ích:** Auto-deploy trên push main, với tagging version.

# 10. Mở Rộng Hệ Thống

## 10.1 Scale Horizontal (Thêm Replicas)

bash

docker service scale student-management\_webapp=5 *# Tăng từ 2 lên 5*

* **Giải Thích:** Swarm tự phân tải qua Nginx. Theo dõi với docker service ps.
* **Giới Hạn:** Cần đủ nodes; traffic cao → thêm auto-scaling script.

**10.2 Scale Vertical (Tăng Tài Nguyên)**

Chỉnh docker-stack.yml:

yaml

deploy:

resources:

limits:

cpus: '0.5'

memory: 512M

reservations:

cpus: '0.25'

memory: 256M

* **Redeploy:** docker stack deploy -c docker-stack.yml student-management.

## 10.3 Thêm Nodes Vào Cluster

bash

*# Trên manager: docker swarm join-token worker*

*# Output token, sau đó trên worker:*

docker swarm join --token <token> <manager-ip>:2377

# 11. Xử Lý Sự Cố (Troubleshooting)

## 11.1 Các Vấn Đề Thường Gặp Và Giải Pháp

| **Vấn Đề** | **Triệu Chứng** | **Giải Pháp Chi Tiết** |
| --- | --- | --- |
| **Service Không Khởi Động** | Replicas 0/2, status "Failed" | 1. docker service logs ... (tìm error e.g., port conflict). 2. docker service ps ... --no-trunc (xem events). 3. Kiểm tra image pull: docker pull <image>. |
| **Không Kết Nối DB** | Flask error 500 | 1. Verify .env vars. 2. Test ping: docker exec webapp ping database. 3. Kiểm tra init-db.sql chạy chưa. |
| **Images Không Tìm Thấy** | Pull failed | 1. Kiểm tra login Docker Hub. 2. Verify tag in stack.yml. 3. Sử dụng docker image ls local. |
| **Cổng Bị Chiếm** | Bind error | 1. `netstat -tuln |
| **Permission Denied** | Docker commands fail | 1. sudo usermod -aG docker $USER; newgrp docker. 2. chmod +x \*.sh cho scripts. |

## 11.2 Quy Trình Debug Tổng Quát

1. Kiểm tra logs đầu tiên.
2. Inspect service/node.
3. Restart: docker service update --force ....
4. Nếu stuck, remove và redeploy stack.

# 12. Kiểm Tra Hệ Thống (Health Checks)

## 12.1 Lệnh Kiểm Tra Cơ Bản

bash

*# Containers status*

docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"

*# DB version và data*

docker exec -it $(docker ps -q -f name=database) psql -U postgres -d students\_db -c "SELECT version(); SELECT COUNT(\*) FROM students;"

*# Web test*

curl -f -X GET http://localhost/ || echo "Web app không phản hồi"

curl -f -X POST http://localhost/login -d "username=admin&password=admin123" *# Test auth*

*# Resources*

docker system df *# Images, containers, volumes size*

## 12.2 Health Check Trong Stack.yml

Thêm vào service definition:

yaml

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:5000/health"]

interval: 30s

timeout: 10s

retries: 3

* **Công Cụ Bổ Sung:** Portainer cho UI management Swarm.