**Hướng dẫn CI/CD với Docker & GitHub Actions**

1. **Yêu cầu trước khi bắt đầu**

* Tài khoản GitHub + repository chứa source code.
* Tài khoản Docker Hub để lưu image
* Server (VPS/Cloud): Ubuntu 20.04/22.04 đã cài sẵn Docker.

1. **Tạo Dockerfile**

Ví dụ ứng dụng .NET9 (bạn hãy thay bằng tech stack của mình để phù hợp với project của bạn):

# Stage 1: Build

FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build

WORKDIR /src

# Copy solution and projects

COPY LearnKing.sln ./

COPY LearnKing.Application/\*.csproj ./LearnKing.Application/

COPY LearnKing.Common/\*.csproj ./LearnKing.Common/

COPY LearnKing.Domain/\*.csproj ./LearnKing.Domain/

COPY LearnKing.Infrastructure/\*.csproj ./LearnKing.Infrastructure/

COPY LearnKing.Api/\*.csproj ./LearnKing.Api/

# Restore

RUN dotnet restore LearnKing.sln

# Copy all source code

COPY . .

# Publish

RUN dotnet publish LearnKing.Api/LearnKing.Api.csproj -c Release -o /app/publish /p:UseAppHost=false

# Stage 2: Runtime

FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS runtime

WORKDIR /app

COPY --from=build /app/publish .

ENV ASPNETCORE\_URLS=http://+:86

ENV DOTNET\_RUNNING\_IN\_CONTAINER=true

ENV DOTNET\_SYSTEM\_GLOBALIZATION\_INVARIANT=1

# Xoá file không cần thiết

RUN find /app -name "\*.pdb" -delete && \

    find /app -name "\*.xml" -delete

EXPOSE 86

# Run ứng dụng

ENTRYPOINT ["dotnet", "LearnKing.Api.dll"]

1. **Viết docker-compose.yml (Tùy chọn)**

Nếu web của bạn cần nhiều service như frontend, backend, database,… thì bạn hãy tạo 1 docker-compose.yml để quản lý, dưới đây là 1 ví dụ với 1 service

version: '3.8'

services:

  backend-learnking:

    build:

      context: ./learnking.server

      dockerfile: Dockerfile

    image: ${DOCKER\_USERNAME}/learnking-backend-net:latest

    container\_name: learnking-backend-net

    environment:

      - ASPNETCORE\_ENVIRONMENT=Production

      - ConnectionStrings\_\_DefaultConnection=${DB\_CONNECTION\_STRING}

    networks:

      - app-network

    restart: always

networks:

  app-network:

    driver: bridge

1. **Push code lên GitHub**

git init

git remote add origin git@github.com:your-username/your-repo.git

git add .

git commit -m "init project with docker"

git push -u origin main

1. **Tạo GitHub Actions Workflow**

Tạo thư mục **.github/workflows/deploy.yml** trong repo. Hãy diều chỉnh để phù hợp vs project của bạn.

name: LearnKing CI/CD Pipeline

on:

  push:

    branches:

      - main

jobs:

  build-and-deploy:

    runs-on: ubuntu-latest

    permissions:

      contents: read

      packages: write

    steps:

      - name: Checkout source code

        uses: actions/checkout@v4

        with:

          fetch-depth: 0

      - name: Install Docker Compose

        run: |

          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

          docker-compose --version

      - name: Set up Docker Buildx

        uses: docker/setup-buildx-action@v3

        with:

          install: true

          version: latest

      - name: Cache Docker layers

        uses: actions/cache@v4

        with:

          path: /tmp/.buildx-cache

          key: ${{ runner.os }}-buildx-${{ github.sha }}

          restore-keys: |

            ${{ runner.os }}-buildx-

      - name: Login to Docker Hub

        uses: docker/login-action@v3

        with:

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

          password: ${{ secrets.DOCKER\_PASSWORD }}

      - name: Build and push Docker images

        env:

          DOCKER\_BUILDKIT: 1

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

          DB\_CONNECTION\_STRING: ${{ secrets.DB\_CONNECTION\_STRING }}

        run: |

          # Build images với BuildKit

          docker-compose -f docker-compose.yml build \

            --build-arg BUILDKIT\_INLINE\_CACHE=1

          # Push images

          docker-compose -f docker-compose.yml push backend-learnking

        continue-on-error: false

      - name: Deploy to production server

        uses: appleboy/ssh-action@v1.0.3

        env:

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

          DOCKER\_PASSWORD: ${{ secrets.DOCKER\_PASSWORD }}

        with:

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

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

          key: ${{ secrets.SERVER\_SSH\_KEY }}

          script: |

            cd LearnKing

            ls -la

            # Write docker-compose file

            echo "${{ secrets.DOCKER\_COMPOSE\_FILE }}" > docker-compose.yml

            # Verify docker-compose file

            docker-compose config || {

              echo "Error in Docker Compose file:";

              docker-compose config;

              exit 1;

            }

            # Login to Docker Hub

            echo "${DOCKER\_PASSWORD}" | docker login -u "${DOCKER\_USERNAME}" --password-stdin

            # Pull new images

            docker-compose pull

            # Stop and remove old containers if they exist\

            docker ps -a --filter "name=leaarking-backend-net" --format '{{.ID}}' | xargs -r docker stop || true

            docker ps -a --filter "name=leaarking-backend-net" --format '{{.ID}}' | xargs -r docker rm || true

            # Start containers

            docker-compose up -d

            # Clean up old images

            docker image prune -f

            # Wait for containers to start and verify deployment

            timeout 60 bash -c '

              until docker-compose ps | grep "Up"; do

                echo "Waiting for containers to start...";

                sleep 5;

              done

            ' || exit 1

      - name: Cleanup

        if: always()

        run: |

          docker logout

          rm -rf /tmp/.buildx-cache

* **Giải thích file GitHub Actions: LearnKing CI/CD Pipeline**
* *name: LearnKing CI/CD Pipeline* : Đặt tên cho workflow. Nó sẽ hiển thị trên tab Actions của GitHub.
* Trigger workflow: Workflow chạy mỗi khi có code được push vào nhánh main.

on:

push:

branches:

- main

* Khai báo Job chính

jobs:

build-and-deploy:

runs-on: ubuntu-latest

permissions:

contents: read

packages: write

runs-on: ubuntu-latest → GitHub sẽ tạo 1 runner Ubuntu mới để thực hiện job.

permissions → cấp quyền:

contents: read: đọc nội dung repo.

packages: write: push image Docker lên Docker Hub (hoặc GHCR).

* Các bước trong Job

**Bước 1: Checkout code**

- name: Checkout source code

uses: actions/checkout@v4

with:

fetch-depth: 0

Lấy toàn bộ source code từ repo về runner. fetch-depth: 0 giúp lấy full history (quan trọng nếu cần git logs).

**Bước 2: Cài Docker Compose**

- name: Install Docker Compose

run: |

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

docker-compose --version

GitHub runner mặc định có Docker nhưng không phải lúc nào cũng có Docker Compose → bạn cài thủ công bản 2.20.0.

**Bước 3: Set up Docker Buildx**

- name: Set up Docker Buildx

uses: docker/setup-buildx-action@v3

with:

install: true

version: latest

Cấu hình Docker Buildx (cho phép build multi-platform images, caching hiệu quả hơn).

**Bước 4: Cache Docker layers**

- name: Cache Docker layers

uses: actions/cache@v4

with:

path: /tmp/.buildx-cache

key: ${{ runner.os }}-buildx-${{ github.sha }}

restore-keys: |

${{ runner.os }}-buildx-

Dùng cache để tăng tốc build image (tránh build lại từ đầu mỗi lần).

**Bước 5: Login Docker Hub**

- name: Login to Docker Hub

uses: docker/login-action@v3

with:

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

password: ${{ secrets.DOCKER\_PASSWORD }}

Đăng nhập Docker Hub bằng GitHub Secrets.

**Bước 6: Build & Push Docker image**

- name: Build and push Docker images

env:

DOCKER\_BUILDKIT: 1

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

DB\_CONNECTION\_STRING: ${{ secrets.DB\_CONNECTION\_STRING }}

run: |

# Build images với BuildKit

docker-compose -f docker-compose.yml build \

--build-arg BUILDKIT\_INLINE\_CACHE=1

# Push images

docker-compose -f docker-compose.yml push backend-learnking

Quy trình:

* Build image backend-learnking từ docker-compose.yml
* Push image lên Docker Hub

**Bước 7: Deploy lên server (Production)**

- name: Deploy to production server

uses: appleboy/ssh-action@v1.0.3

env:

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

DOCKER\_PASSWORD: ${{ secrets.DOCKER\_PASSWORD }}

with:

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

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

key: ${{ secrets.SERVER\_SSH\_KEY }}

script: |

cd LearnKing

ls -la

# Write docker-compose file

echo "${{ secrets.DOCKER\_COMPOSE\_FILE }}" > docker-compose.yml

# Verify docker-compose file

docker-compose config || {

echo "Error in Docker Compose file:";

docker-compose config;

exit 1;

}

# Login to Docker Hub

echo "${DOCKER\_PASSWORD}" | docker login -u "${DOCKER\_USERNAME}" --password-stdin

# Pull new images

docker-compose pull

# Stop and remove old containers

docker ps -a --filter "name=leaarking-backend-net" --format '{{.ID}}' | xargs -r docker stop || true

docker ps -a --filter "name=leaarking-backend-net" --format '{{.ID}}' | xargs -r docker rm || true

# Start containers

docker-compose up -d

# Clean up old images

docker image prune -f

# Wait until containers start

timeout 60 bash -c '

until docker-compose ps | grep "Up"; do

echo "Waiting for containers to start...";

sleep 5;

done

' || exit 1

Các bước khi deploy trên server:

* cd LearnKing → vào thư mục chứa app
* Ghi file docker-compose.yml từ secret (DOCKER\_COMPOSE\_FILE)
* Validate file docker-compose config
* Login Docker Hub
* Pull image mới
* Stop & remove container cũ leaarking-backend-net
* docker-compose up -d → chạy container mới
* Xoá image cũ (docker image prune -f)
* Đợi tối đa 60s cho container start thành công

**Bước 8: Cleanup**

- name: Cleanup

if: always()

run: |

docker logout

rm -rf /tmp/.buildx-cache

Luôn chạy (kể cả khi lỗi): logout Docker Hub và xoá cache buildx để runner sạch sẽ.

* **Quy trình CI/CD tổng thể**
* Dev push code lên main
* GitHub Actions:
* Checkout code
* Build Docker image từ docker-compose.yml
* Push image lên Docker Hub
* Deploy step:
* SSH vào server
* Pull image mới về
* Stop & remove container cũ
* docker-compose up -d → chạy container mới
* Cleanup + Verify → nếu container chưa Up trong 60s thì fail pipeline.

1. **Cấu hình GitHub Secrets**

Vào GitHub Repo → Settings → Secrets and variables → Actions → New repository secret

Thêm các key sau:

* DOCKER\_USERNAME → username Docker Hub
* DOCKER\_PASSWORD → password Docker Hub
* SERVER\_HOST → IP VPS (ví dụ: 123.45.67.89)
* SERVER\_USER → user SSH (ví dụ: ubuntu hoặc root)
* SERVER\_SSH\_KEY → Private SSH Key (nội dung file ~/.ssh/id\_rsa)
* Trên VPS, nhớ thêm Public Key vào ~/.ssh/authorized\_keys.
* DB\_CONNECTION\_STRING → chuỗi kết nối database (nếu cần), ỏ đây mình thêm chuỗi kết nối vào GitHub Secrets để tăng tính bảo mật
* DOCKER\_COMPOSE\_FILE → file docker compose để pull image chúng ta dẩy lên docker hub về server của chúng ta, dưới đây là 1 ví dụ bạn hãy chỉnh sửa lại để phù hợp với project của bạn.

version: '3.8'

services:

backend:

image: truongdev1510/learnking-backend-net:latest

container\_name: learnking-backend-net

ports:

- 5005:86

environment:

- ASPNETCORE\_ENVIRONMENT=Production

- ConnectionStrings\_\_DefaultConnection=<thay bằng chuỗi kết nối của bạn>

- DOTNET\_SYSTEM\_GLOBALIZATION\_INVARIANT=0

networks:

- app-network

restart: always

healthcheck:

test: [ CMD, curl, -f, http://localhost:86/health ]

interval: 30s

timeout: 10s

retries: 3

start\_period: 10s

volumes:

- ./keys:/root/.aspnet/DataProtection-Keys

networks:

app-network:

driver: bridge

A computer screen with a computer screen

Description automatically generated

1. **Kiểm tra pipeline**

* Push code mới lên main:

git add .

git commit -m "update app"

git push

* Vào GitHub → tab Actions → xem workflow chạy.
* Quy trình:
* Build Docker image → push lên Docker Hub.
* SSH vào server.
* Pull image mới.
* Stop container cũ.
* Run container mới.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Kiểm tra trên server**

* SSH vào server:

docker ps

* Sẽ thấy container chạy.
* Mở trình duyệt: http://<SERVER\_HOST>:<port>

**A screenshot of a computer

Description automatically generated**