**CI/CD Jenkins + Docker + GitHub (Step-by-Step)**

**1. Yêu cầu cài đặt & Chuẩn bị**

**1.1. Hệ điều hành**

* Ubuntu Server 22.04 LTS (khuyến nghị)
* RAM tối thiểu 2GB (tốt hơn là 4GB nếu build .NET/NodeJS nặng)
* Disk ≥ 30GB

**1.2. Tài khoản & công cụ**

* GitHub: nơi lưu source code.
* Docker Hub: nơi push/pull image.
* Server production: nơi deploy (cài Docker + Compose).
* Jenkins server: có thể cùng hoặc tách với production.1.3. Network

**1.3. Network**

Mở port:

* 8080 → Jenkins.
* 22 → SSH.
* 80/443 → Web app.

**1.4. Cấu trúc source code**

Lưu ý đây là cấu trúc của 1 dự án .NET9

A screenshot of a computer

Description automatically generated

**2. Cài Jenkins & Setup lần đầu**

**2.1. Cài đặt Jenkins**

# Update hệ thống

sudo apt update && sudo apt upgrade -y

# Cài Java (Jenkins yêu cầu Java 17+)

sudo apt install fontconfig openjdk-21-jre

# Thêm key và repo Jenkins

sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

# Cài Jenkins

sudo apt update

sudo apt install jenkins

# Start Jenkins

sudo systemctl enable jenkins

sudo systemctl start jenkins

**2.2. Truy cập giao diện web**

* Mở trình duyệt: http://<server-ip>:8080
* Lấy admin password: sudo cat /var/lib/jenkins/secrets/initialAdminPassword
* Làm theo wizard: tạo admin user, cài plugin mặc định.

A screenshot of a computer

Description automatically generated

**3. Cài Docker & Docker Compose**

**3.1. Cài Docker**

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

**3.3. Cài docker compose**

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

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

docker-compose --version

**3.4. Thêm quyền cho Jenkins & user**

sudo usermod -aG docker $USER

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

Kiểm tra Docker chạy: docker run hello-world

A screen shot of a computer

Description automatically generated

**4. 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"]

5. **Viết docker-compose.prod.yml cho production server**

version: "3.9"

services:

  api:

    image: docker.io/truongdev1510/server-lms-net:latest

    container\_name: lms-api

    restart: always

    ports:

      - "3007:86"

    environment:

      ASPNETCORE\_ENVIRONMENT: Production

      ConnectionStrings\_\_DefaultConnection: ${DB\_CONNECTION\_STRING}

      DOTNET\_SYSTEM\_GLOBALIZATION\_INVARIANT: 0

    env\_file:

      - .env

networks:

  app-network:

    driver: bridge

**6. Kết nối GitHub → Jenkins**

**6.1. Tạo PAT (Personal Access Token) trên GitHub**

Vào Settings → Developer settings → Personal access tokens → Fine-grained tokens.

Scope: repo (read), admin:repo\_hook.

Copy token.

A screenshot of a computer

Description automatically generated

**6.2. Add vào Jenkins**

Vào Manage Jenkins → Credentials → Global.

Add new credential:

Kind: Username with password

Username: GitHub username

Password: PAT token

ID: github-pat

A screenshot of a login form

Description automatically generated

**6.3. Tạo webhook GitHub**

Repo → Settings → Webhooks → Add webhook.

URL: http://<jenkins-server>:8080/github-webhook/.

Content type: application/json.

Trigger: push events.

A screenshot of a computer

Description automatically generated

**7. Quản lý Credentials trong Jenkins**

* dockerhub-cred → Docker Hub username + Access Token.
* server-ssh-key → SSH Private Key login vào production.
* db-conn → Secret text chứa connection string.
* docker-compose-file --> docker file production

A screenshot of a computer

Description automatically generated

**8. Chuẩn bị Production server**

Cài Docker + Docker Compose.

Copy public key vào ~/.ssh/authorized\_keys.

Có sẵn thư mục ~/project/

**9. Jenkinsfile (Pipeline Script)**

pipeline {

    agent any

    environment {

        REGISTRY = "docker.io/${DOCKER\_USERNAME}"

        IMAGE\_NAME = "server-lms-net"

        SERVER\_HOST = "103.20.96.174"

        SERVER\_USER = "root"

    }

    stages {

        stage('Checkout') {

            steps {

                checkout([$class: 'GitSCM',

                  branches: [[name: '\*/master']],

                  userRemoteConfigs: [[

                    url: 'https://github.com/XT-xuantruong/learnking.server.git',

                    credentialsId: 'github-pat'

                  ]]

                ])

            }

        }

        // stage('Build & Test') {

        //     steps {

        //         sh 'dotnet restore LearnKing.sln'

        //         sh 'dotnet build LearnKing.sln -c Release'

        //         //sh 'dotnet test LearnKing.sln'

        //     }

        // }

        stage('Docker Build') {

            steps {

                withCredentials([usernamePassword(credentialsId: 'dockerhub-cred',

                    usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {

                    sh "docker build -t docker.io/$DOCKER\_USER/$IMAGE\_NAME:latest ."

                }

            }

        }

        stage('Push Docker Hub') {

            steps {

               withCredentials([usernamePassword(credentialsId: 'dockerhub-cred',

                    usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {

                    sh "echo $DOCKER\_PASS | docker login -u $DOCKER\_USER --password-stdin"

                    sh "docker push docker.io/$DOCKER\_USER/$IMAGE\_NAME:latest"

                }

            }

        }

        stage('Deploy Server') {

            steps {

                withCredentials([

                    usernamePassword(credentialsId: 'dockerhub-cred',

                        usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS'),

                    string(credentialsId: 'db-conn', variable: 'DB\_CONN'),

                    file(credentialsId: 'docker-compose-file', variable: 'DOCKER\_COMPOSE\_PATH')

                ]) {

                    sshagent (credentials: ['server-ssh-key']) {

                        sh '''

                        # Copy docker-compose.yml từ Jenkins sang server

                        scp -o StrictHostKeyChecking=no $DOCKER\_COMPOSE\_PATH $SERVER\_USER@$SERVER\_HOST:~/project/docker-compose.yml

                        # SSH vào server để deploy

                        ssh -o StrictHostKeyChecking=no $SERVER\_USER@$SERVER\_HOST "

                        cd ~/project && \

                        echo \\"DB\_CONNECTION\_STRING=$DB\_CONN\\" > .env && \

                        echo \\"$DOCKER\_PASS\\" | docker login -u $DOCKER\_USER --password-stdin && \

                        docker compose --env-file .env pull && \

                        docker compose --env-file .env down && \

                        docker compose --env-file .env up -d && \

                        docker image prune -f

                        "

                        '''

                    }

                }

            }

        }

    }

}

**10. Tạo Jenkins Pipeline Job**

* Dashboard → New Item → Pipeline → đặt tên LMS-CICD.
* Pipeline script from SCM → Git → repo URL + branch master → credentials github-pat.
* Save.

A screenshot of a computer

Description automatically generated

**11. Test Pipeline**

* Push code lên GitHub.
* Jenkins trigger job → chạy các stage: Checkout → Build → Docker Build → Push → Deploy.
* A screenshot of a computer

  Description automatically generated

A screenshot of a computer

Description automatically generated

* Vào server check container: docker ps
* Mở http://<server-ip>:<app-port> để test app.

A screenshot of a computer

Description automatically generated