# FINAL PROJECT BRIEF — DEVOPS, CI/CD & CONTAINERIZATION

**COURSE: 020IDCE5 - INTÉGRATION ET DÉPLOIEMENT CONTINUE**

**EVALUATION DATE: DECEMBER 22**
**GROUP WORK: 2–3 STUDENTS**
**TOTAL POINTS: 100 (+10 BONUS)**
**INSTRUCTORS: NADIM HENOUD, TOUFIC FAKHRY**

## Contents

## 1. Project Overview

In this final project, your team will design, containerize, automate, and deploy a small application using DevOps principles and tooling.
You may choose any simple application or template, provided it is multi-service and includes a mandatory database.
The project evaluates your understanding of:

- Docker and containerization
- Docker Compose orchestration
- CI/CD workflows using GitHub Actions or GitLab CI
- Kubernetes deployment
- Persistent data management
- DevOps architectural reasoning
- Ability to justify and defend technical decisions

You must design, implement, document, and defend your technical choices during a live evaluation.

## 2. Project Requirements

### 2.1 Application (Mandatory Persistence)

Your application must include:

- A backend/API service
- A database service (PostgreSQL recommended; MySQL, MariaDB, or MongoDB acceptable)
- At least one feature that reads and writes data

This ensures a clear demonstration of persistent state management.

### 2.2 Docker and Containerization

Each service must define a clean, production-ready Dockerfile using best practices:

- Lightweight base image
- Proper working directory
- .dockerignore
- Clear dependency installation
- Exposed ports
- Functional entrypoint/CMD
- Consistent builds

Evaluation includes correctness, clarity, image quality, and reproducibility.

### 2.3 Docker Compose Stack

Your Compose setup must orchestrate:

- Backend service
- Database service
- Persistent volumes
- Networks for communication
- Environment variables for credentials
- Service startup ordering (depends_on or healthchecks)

The entire stack must launch with one command.

## 2.4 CI/CD Pipeline

### Create an automated workflow using GitHub Actions or GitLab CI.

Mandatory stages:

1. **Test stage**
2. **Build stage**
3. **Registry push stage** (Docker Hub or GitLab/GitHub registry)

Evaluation includes:

- Pipeline correctness
- Use of secrets and environment variables
- Proper triggers and conditions
- Demonstrable execution in your repository

## 2.5 Kubernetes Deployment

Deploy your application to a local Kubernetes cluster (Docker Desktop K8s, Minikube, or Kind).

### Your deployment must include:

- Deployment resource
- Service (NodePort or ClusterIP)
- ConfigMap or Secret
- Readiness and liveness probes
- Demonstrable access via Kubernetes
- **Database location requirement (choose one):**

  #### Option A: Database deployed inside Kubernetes

  - StatefulSet
  - PersistentVolumeClaim
  - Headless service

  #### Option B: External database (Compose or cloud)

  - Requires clear written justification
  - Must connect correctly and securely

## 3. Technical Defense (Mandatory)

Each team must complete a **live 10-minute technical defense**.

### Evaluation criteria:

- **Architectural justification**
  Ability to explain the reasoning behind architectural and design decisions (images, container structure, cluster layout, database placement).
- **CI/CD pipeline reasoning**
  Clear explanation of pipeline stages, variable handling, rules/triggers, and artifacts.
- **Kubernetes reasoning**
  Understanding of Deployments, Services, probes, pods, scaling behavior, and how stateful components were handled.

This component measures **understanding**, not memorization.

## 4. Optional Bonus (up to +10 pts)

Choose one (or more):

- **Monitoring** (Prometheus, Grafana, metrics endpoint)
- **Infrastructure-as-Code automation** (Ansible or Terraform)
- **Advanced CI/CD logic** (multi-environment, rolling strategies, tagging systems)

## 5. Deliverables

1. **Git repository**, including:
    - Source code
    - Dockerfiles
    - docker-compose.yml
    - CI/CD configuration
    - Kubernetes manifests
    - README.md
2. **README.md** containing:
    - Overview
    - Architecture diagram
    - Compose and CI/CD instructions
    - Kubernetes deployment instructions
    - Database design justification
3. **Technical demonstration and oral defense (mandatory)**
   A live walkthrough of pipeline, containerization, deployment, and design choices.

## 6. Evaluation Rubric

| Component | Points |
|---|---|
| **Application functionality** | 10 |
| **Dockerfiles** | 15 |
| **Docker Compose** | 15 |
| **CI/CD pipeline** | 25 |
| **Kubernetes deployment** | 20 |
| **Technical Defense** | **15** |
| **Bonus** | +10 |

**Raw total: 110 points**
Final grade will be scaled to **100 points**.

## 7. Final Notes

- Keep the application simple; focus on DevOps implementation.
- Begin early; pipeline and Kubernetes debugging require time.
- Ensure documentation is clear, structured, and complete.
- Be prepared to fully justify all decisions during the defense session.