# Laboratoire 0 - Infrastructure - Docker, Kubernets, Version Control, CD/CI

Cours : **Architecture Logicielle (LOG430)**

Session : **Été 2025**

Date du laboratoire : **Semaine du 12 mai 2025**


Le génie pour l'industrie

# Type de laboratoire

**Préparatoire**

# 1. Contexte

Dans tout projet logiciel structuré, la capacité à travailler dans un environnement reproductible, automatisé et bien organisé est essentielle. Avant de s'engager dans l'étude des styles architecturaux et des principes de conception, il est important d'établir un socle technique fiable pour expérimenter ces concepts.

Ce laboratoire introductif vous invite à créer un environnement de développement **conteneurisé**, **versionné** et **testé automatiquement** en utilisant les outils de votre choix. Ce socle sera réutilisé et enrichi dans les laboratoires suivants, lors de l'implémentation d'architectures logicielles plus complexes.

**Important :** Ce laboratoire fait partie de l'**Étape 1 de l'évaluation**, en combinaison avec les laboratoires 1 et 2. Il est donc évalué, et les livrables produits feront partie du livrable final attendu à la fin de l'Étape 1.

# 2. Objectifs d'apprentissage

À l'issue de ce laboratoire, vous serez capable de :

- Structurer un dépôt de projet dans un gestionnaire de code source (GitLab ou GitHub)
- Mettre en place des tests unitaires automatisés
- Conteneuriser l'application avec Docker
- Automatiser les étapes de vérification, test et build avec une pipeline CI/CD

# 3. Pré-requis

Avant de commencer, assurez-vous de disposer de :

- Un compte personnel sur **GitLab** ou **GitHub**
- Un éditeur de code (VSCode recommandé)
- Les outils suivants installés sur votre machine virtuelle :
  - Git
  - Docker
  - Un framework de test adapté à votre langage (ex : `pytest`, `JUnit`, `Jest`)

### Accès aux machines virtuelles

Chaque étudiant recevra un accès individuel à une machine virtuelle dédiée pour réaliser les travaux pratiques de ce laboratoire. Ces machines sont préconfigurées avec les outils nécessaires (Docker, Git, etc.).

- Les **identifiants de connexion (adresse IP, nom d'utilisateur, mot de passe)** seront communiqués **sur le serveur Discord du cours**.
- Il est de votre responsabilité de tester votre accès avant la séance de laboratoire.
- En cas de problème, veuillez contacter rapidement le chargé de laboratoire.

# 4. Contenu du laboratoire

## 4.1. Création et structuration du dépôt

- Créez un dépôt privé nommé sur GitLab ou GitHub.
- Ajoutez un fichier `README.md` contenant :
  - Une brève description de l'application
  - Les instructions d'exécution
  - La structure du projet
- Ajoutez un fichier `.gitignore` adapté à votre environnement.

## 4.2. Développement d'une application minimale

Le but de ce laboratoire est de mettre en plac mais de mettre en place un socle technique structuré, autour d'une application très simple. Le contenu fonctionnel peut se limiter à un simple affichage d'un message texte Hello World, ou une réponse minimale sur un port web.

- Créez une application simple dans le langage de votre choix (Python, Java, JavaScript, etc.)
- L'application peut Afficher un message dans la console.

## 4.3. Tests unitaires automatisés

- Rédigez au moins deux tests unitaires.
- Utilisez un framework adapté (ex. `pytest`, `JUnit`, `Jest`).

## 4.4. Conteneurisation avec de l'application

- Créez une image Docker pour votre API.
- Vérifiez que l'application fonctionne dans un conteneur local.

### 4.5. Orchestration avec Docker Compose

Docker Compose est un outil essentiel pour centraliser et automatiser le lancement des conteneurs au sein d'un même environnement de développement. Dans le cadre de ce laboratoire, il permet de lancer votre application.

**Remarque :** Vous pouvez également explorer et utiliser des alternatives compatibles pour la conteneurisation comme **podman** ou **containerd**, si vous êtes déjà familiers avec ces outils ou souhaitez approfondir vos connaissances.

### 4.6. Intégration continue (CI/CD)

L'intégration continue vise à automatiser les étapes clés de vérification du code à chaque modification du projet. Cela permet de détecter les erreurs rapidement, de garantir la qualité du code, et de livrer plus fréquemment des versions fonctionnelles et testées.

Vous devez configurer une **pipeline CI/CD** en utilisant **GitLab CI/CD** ou **GitHub Actions**, qui s'exécutera automatiquement à chaque `push` ou `merge request`. La pipeline doit inclure au minimum les trois étapes suivantes, exécutées dans l'ordre suivant :

1. **Lint** : vérifie la qualité syntaxique et stylistique du code via un outil d'analyse statique (ex : `pylint`, `black`, `eslint`). Le linting permet de :
   - Détecter des erreurs de style ou de structure
   - Unifier l'apparence du code
   - Améliorer sa lisibilité et sa maintenabilité
2. **Tests unitaires** : vérifie automatiquement que les fonctionnalités principales fonctionnent. En cas d'échec, la pipeline s'interrompt.
3. **Build** : si les étapes précédentes sont réussies, construisez l'image Docker de l'application.
4. **Publication sur Docker Hub** : à la suite du build, l'image Docker doit être automatiquement poussée sur votre compte Docker Hub personnel. Vous pouvez nommer l'image selon le format `docker.io/username/nom-image:tag`.

**Remarque importante :** Vous êtes fortement encouragés à concevoir une image Docker **légère et optimisée**. Pour cela :

- Utilisez des images de base minimalistes comme `python:3.11-slim`, `alpine`.
- Évitez d'inclure des fichiers inutiles dans l'image (utilisez `.dockerignore`).
- Nettoyez les caches ou dépendances après installation.

## 5. Conseils

- Commencez par une structure de projet simple et améliorez-la progressivement.
- Travaillez en local, testez, puis poussez sur GitLab ou GitHub.
- Utilisez des branches et des messages de commit clairs.
- Documentez vos choix et étapes dans le `README.md`.

# 6. Livrables

Le livrable à remettre sur Moodle est un **rapport au format pdf**. Votre repo GitHub ou Gitlab doit être publique afin que le chargé de laboratoire puisse consulter et évaluer votre travail.

Le dépôt doit inclure un fichier `README.md` clair et complet, qui contiendra obligatoirement les éléments suivants :

- Une présentation de l'architecture et de la structure du projet
- Une explication claire des étapes nécessaires pour :
  - Cloner le projet et se placer dans le bon répertoire
  - Construire et lancer le conteneur à l'aide de Docker Compose
- Une capture d'écran (ou lien) montrant une exécution réussie de la pipeline CI/CD

# 7. Ressources utiles

- Docker – Getting Started
- GitHub Actions – Quickstart
- GitLab CI/CD – Guide
- Guide des tests unitaires