# 研修課題5：CI/CDとGitLabのIssue運用

## 1. CI/CDの概要

CI（継続的インテグレーション）とは、開発者が行った変更を頻繁に統合し、自動でビルド・テストを行う手法。  
CD（継続的デリバリー/継続的デプロイ）は、CIで統合されたコードを本番環境に自動または手動でリリースする仕組み。  
GitLabやGitHubでは、CI/CDパイプラインを用いて自動化を進めることができる。

## 2. パイプライン構成と実践例

一般的なCI/CDパイプラインの構成は以下の通りです：  
・build：ソースコードのビルドやパッケージ作成  
・test：自動テストによる品質確認  
・deploy：開発/本番環境へのデプロイ  
  
例：GitLab CIでの .gitlab-ci.yml  
stages:  
 - build  
 - test  
 - deploy  
  
build-job:  
 stage: build  
 script:  
 - echo "Building..."  
  
test-job:  
 stage: test  
 script:  
 - echo "Testing..."  
  
deploy-job:  
 stage: deploy  
 script:  
 - echo "Deploying..."

## 3. CI/CDのベストプラクティス

・小さく頻繁にコミットし、CIを高速に保つ  
・ビルド時間は10分以内を目安に  
・失敗の原因は個人ではなく仕組みに注目する  
・テスト環境は本番環境とできる限り一致させる  
・パイプラインステージは明確に分ける  
・ドキュメントの整備と社内共有を継続する

## 4. GitLabのIssue運用

GitLabのIssueは、タスク・バグ・機能追加などの管理に使う。  
Markdownで記述可能で、コメント・チェックリスト・ラベル・マイルストーンなどを使って詳細管理ができる。

## 5. 良いIssueの書き方

・明確なタイトルをつける（例：API連携処理のエラーハンドリング追加）  
・本文構成：背景、目的、対応内容、完了条件、参考資料など  
・ラベルやマイルストーンを活用し、優先度や進行状況を明示する  
・チェックリストは簡潔に、複雑なものは別Issue化  
・担当者（Assignee）を明示  
・テンプレートを使用して統一感を保つ

## 6. クイックアクション（便利なショートコマンド）

GitLabではIssueのコメント欄でクイックアクションが利用できます：  
・/assign @user  
・/label ~bug  
・/weight 2  
・/close  
・/duplicate #123  
・/epic #456

## 7. まとめ

CI/CDは現代的な開発フローにおいて必須の仕組みであり、品質と開発スピードを両立する手段。  
GitLabではCI/CDだけでなく、Issueを活用したチーム内のタスク管理も効率化できる。