# Gitlab et Gitlab-CI

Git est le système de contrôle de version distribué qui fait aujourd'hui l'unanimité quant à son efficacité et les réponses apportées aux besoins du développement collaboratif.

La plate-forme GitLab a permis d'intégrer Git au sein d'une plate-forme complète pour la gestion de projet, du développement jusqu'au dépoiement. Gitlab facilite la gestion des dépôts et la mise en place de workflows de collaboration éprouvés entre développeurs, reviewers, chefs de projets et administrateurs.

Mais ses fonctionnalités ne s'arrêtent pas là. C'est également une solution complète pour l'intégration continue et le développement continu. Associé à des technologies comme la contenerisation avec Docker, il devient de plus en plus incontournable pour les développeurs, les administrateurs système, ...

## Objectifs pédagogiques

- Utiliser Gitlab pour reccueillir et gérer les dépots git
- Connaissance des différents workflows de collaboration
- Gérer un workflow de branches grâce aux merge requests
- Mise en place d'intégration continue et déploiement continu

## Durée

3 jours (21 heures)


### Contact

Anne NICOLAS – anicolas@ossflow.fr
+33 6 59 11 75 55
https://ossflow.fr/formation

# Plan de formation

JOUR 1

## Maîtriser Git et la gestion de projet avec GitLab

### 1 - Rappel des fondamentaux du fonctionnement de Git

- Comprendre les objets de Git (commit, arbre, tag, branche).
- Fonctionnement des branches et gestion avancée des tags.
- Différences entre tag léger et tag annoté.
- Focus sur les commandes spécifiques GitLab :
  - Cherry-pick : appliquer des commits spécifiques.
  - Revert : annuler proprement des modifications.

### 2 - GitLab : Optimiser la gestion de code

- Présentation des différents workflows Git (feature branch, GitFlow, trunk-based).
- Utilisation de l'interface GitLab pour gérer les dépôts :
  - Navigation dans l'interface.
  - Outils complémentaires (Web IDE, éditeur de fichiers, recherches avancées).
- Organisation des projets :
  - Releases et gestion des milestones.
  - Utilisation et structuration des groupes.
- Gestion des permissions :
  - Différents niveaux d'accès.
  - Bonnes pratiques de sécurité.

JOUR 2

## Collaboration avancée avec GitLab

- Utilisation avancée des Merge Requests :
  - Approvisionnement, révision, stratégies de merge.
- Gestion des issues et des templates :
  - Structuration des tickets.
  - Utilisation de modèles pour homogénéiser les demandes.
- Organisation avec labels et issue boards :
  - Gestion visuelle des tâches et workflows agiles.

*Atelier pratique : Création complète d'un projet GitLab : mise en place du dépôt, des workflows de merge requests, des issues et de la roadmap.*

## Maîtriser GitLab-CI pour l'intégration et le déploiement

### 1 - GitLab-CI : Mise en place de l'intégration et du déploiement continu

- Architecture de GitLab-CI : runners, orchestrateurs, exécutants.
- Concepts clés : Pipelines, stages, jobs.
- Rédaction et personnalisation du fichier .gitlab-ci.yml :
  - Structure.
  - Définition des pipelines simples et multi-stage
  - Utilisation des variables
  - Définitoin des conditions d'exécution

JOUR 3

### 2 - Aller plus loin sur la CI/CD

- Optimiser l'usage de Docker avec GitLab-CI :
  - Images de build.
  - Gestion des artefacts et caches.
- Concepts avancés :
  - Stratégies de parallélisation.
  - Déploiement conditionnel
  - Triggers de pipelines
- Développer une CI/CD maintenable
  - Factoriser le contenu : extends et ancres
  - Splitter, mutualiser : include
  - Catalogue CI/CD
- Les spécificités liées à Docker
- Déploiements continus : les environnements
- Usages avancés de la CI/CD :
  - Tests unitaires
  - Construction et déploiement f'images
  - Gitlab pages

*Atelier pratique : mise en place d'une chaÎne complète CI/CD ainsi que d'un référentiel commun*