Formation **GitLab CI/CD**

L'automatisation des processus de développement est devenue indispensable. L'outil GitLab CI/CD s'est imposé comme une alternative solide à Jenkins, et sa maîtrise devient un atout majeur. Ce programme de formation complet vous accompagnera pas à pas, de la configuration de base à la mise en œuvre de pipelines avancés pour des applications réelles. Vous apprendrez à automatiser les tests, les builds, les déploiements et la gestion des microservices, le tout en utilisant les meilleures pratiques de l'industrie. Gagnez en productivité et en efficacité grâce à notre formation GitLab CI/CD !

Durée

3 jours

Objectifs pédagogiques

- Comprendre les principes fondamentaux de la CI/CD
- Configurer un environnement GitLab CI/CD pour exécuter des pipelines adaptés aux besoins des projets
- Automatiser les processus de build, test et déploiement
- Concevoir des pipelines CI/CD optimisés et maintenables
- Orchestrer la livraison continue de microservices
- Déployer des applications sur des infrastructures cloud (AWS, Kubernetes)

Public

Développeurs, ingénieurs DevOps, architectes, administrateurs

Prérequis

Connaissances de base en Git (création de branches, commits, merges). Notions en administration Linux et scripting shell. Connaissance de Docker. Connaissances de base sur Kubernetes recommandées.

Programme de formation

Phase d'inclusion

Accueil des participants, présentation des objectifs et contextes professionnels de chacun.

Introduction à la formation GitLab : prérequis et configuration de base

Présentation générale de cette formation GitLab CI/CD, approche et objectifs pédagogiques
Configuration des outils nécessaires à la formation

Introduction à la CI/CD et à GitLab CI/CD

Qu'est-ce que la CI/CD ?
Introduction à GitLab CI/CD : pourquoi l'utiliser ?
Comparaison entre GitLab CI/CD et les autres solutions du marché (Jenkins, GitHub Actions, etc.)

Concepts clés de GitLab CI/CD

Présentation des concepts de base
Les jobs : le cœur de la CI/CD
Exécution d'un pipeline complet
Les stages : grouper et organiser les jobs
Needs : gérer les dépendances entre jobs
Script : commandes shell et scripts d'exécution
Only : quand exécuter un job (branches, tags, etc.)
Workflow Rules : gestion avancée du comportement des pipelines
Déclencher un pipeline via une merge request
Utiliser les variables prédéfinies de GitLab
Créer et utiliser des variables personnalisées

Architecture de GitLab CI/CD

Qu'est-ce qu'un GitLab Runner et son rôle ?
GitLab Executors : les différents types d'exécuteurs (Shell, Docker, etc.)
Le flow d'exécution des jobs dans un pipeline
Utilisation de l'exécuteur Docker
Specific Runners : configurer des runners spécifiques à des projets
Démo : configuration d'un runner managé
Installer et enregistrer un runner local sur MacOS
Installer et enregistrer un runner local sur Windows
Prérequis pour AWS
Installer et enregistrer un runner sur AWS EC2
Exécuter un job sur un runner spécifique (tags)
Ajouter un runner Docker sur AWS EC2
Group Runners : partager des runners entre plusieurs projets
Gestion de GitLab Self-Managed
Compatibilité des versions des runners et de GitLab
Récapitulatif de l'architecture GitLab CI/CD

Construire un pipeline CI/CD réel pour une application Node.js

Présentation du projet Node.js
Exécuter des tests unitaires et collecter les rapports de tests
Build d'une image Docker et push dans une registry privée
Déploiement sur un serveur de développement
Introduction aux GitLab Environments (pour suivre où le code est déployé)
Déploiement avec Docker Compose

Optimiser le pipeline CI/CD & Multi-Stage Pipelines

Introduction à l'optimisation des pipelines
Configuration dynamique des versions pour les images Docker
Configurer un cache pour accélérer les exécutions de pipeline
Tests de sécurité (SAST) et vérification de la qualité du code
Introduction au déploiement multi-stage - Démo multi-stage 1
Déploiement en staging - Démo multi-stage 2
Réutilisation des configurations avec extends - Démo multi-stage 3

Déploiement en production - Démo multi-stage 4

Pipelines CI/CD pour des applications Microservices (monorepo & polyrepo)

Qu'est-ce qu'un microservice ?
Monorepo vs Polyrepo : quelle approche choisir ?
Démonstration : exemple concret de monorepo et polyrepo
Démarrer le service en local (Monorepo) et se familiariser avec l'application
Préparation au déploiement (Démo Monorepo 1)
Build des microservices (Démo Monorepo 2)
Déploiement des microservices (Démo Monorepo 3)
CI/CD pour une application en polyrepo
Extraction de la logique commune (Job Templates 1)
Création d'une bibliothèque de templates CI/CD (Job Templates 2)

Déployer des Microservices sur un cluster Kubernetes

Introduction au déploiement sur Kubernetes
Création d'un utilisateur GitLab avec des permissions restreintes
Déploiement sur Kubernetes (partie 1)
Déploiement sur Kubernetes (partie 2)
Conclusion et nettoyage des ressources

Moyens et méthodes pédagogiques

- La formation alterne entre présentations des concepts théoriques et mises en application à travers d'ateliers et exercices pratiques (hors formation de type séminaire).
- Les participants bénéficient des retours d'expérience terrains du formateur ou de la formatrice
- Un support de cours numérique est fourni aux stagiaires

Modalités d'évaluation

- **En amont de la session de formation**, un questionnaire d'auto-positionnement est remis aux participants, afin qu'ils situent leurs connaissances et compétences déjà acquises par rapport au thème de la formation.

- **En cours de formation**, l'évaluation se fait sous forme d'ateliers, exercices et travaux pratiques de validation, de retour d'observation et/ou de partage d'expérience, en cohérence avec les objectifs pédagogiques visés.

- **En fin de session**, le formateur évalue les compétences et connaissances acquises par les apprenants grâce à un questionnaire reprenant les mêmes éléments que l'auto-positionnement, permettant ainsi une analyse détaillée de leur progression.