

## Formation Git : Gestion de dépôts + Gitlab-CI / Intégration continue

<b>Durée :</b>	5 jours
<b>Public :</b>	Opérationnels, Développeurs, Chefs de projets
<b>Pré-requis :</b>	Connaissance du cycle de vie d'une application, maîtrise des commandes de base Git
<b>Objectifs :</b>	Maîtriser l'usage de commandes Git pour la gestion d'un dépôt de sources - Mettre en oeuvre et exploiter un serveur d'intégration continue. Gérer les interconnexions avec un système de build et de tests
<b>Sanction :</b>	Attestation de fin de stage mentionnant le résultat des acquis
<b>Taux de retour à l'emploi:</b>	Aucune donnée disponible
<b>Référence:</b>	DEV101774-F
<b>Note de satisfaction des participants:</b>	Pas de données disponibles

### Découvrir Git

Principes de gestion de contrôle de source (SCM)  
Historique, contrôle local, centralisé et distribué  
Fonctionnement des instantanés, comparaison avec les différences  
Installation (Linux, MacOS, Windows)  
Accès au manuel : man / help  
Configuration initiale de Git : préférences, profil utilisateur  
Initialisation d'un dépôt local

### Atelier : Installation de Git - Création d'un projet

### Comprendre le cycle de vie du répertoire de travail

Concepts de répertoire de travail, index et dépôt  
Vérifier l'état de la copie de travail : status  
Indexer ses modifications : add  
Ignorer des fichier : .gitignore  
Valider ses modifications : commit  
Supprimer et déplacer des fichiers

### Atelier : contributions et validations

### Visualiser l'historique

- Visualiser les modifications : log
- Personnaliser le format : stat, pretty, ...
- Filtrer par date, auteur, message de commit, contenu modifié, ...
- Visualiser et exporter une différence (format natif, outil externe)
- Étiqueter ses validations : étiquettes légères et annotées
- Rechercher avec git-grep

## **Annuler des actions**

- Réécrire la dernière validation
- Désindexer un fichier
- Réinitialiser un fichier

## **Travailler avec les branches**

- Principe de branche, le pointeur HEAD
- Créer une branche
- Basculer entre les branches, le mode détaché
- Fusionner les branches : avance-rapide, trois sources
- Gérer les conflits de fusion
- Outil de fusion externe : mergetool (emerge, vimdiff, meld, ...)
- Visualiser les branches existantes, celles qui ont été fusionnées
- Supprimer une branche
- Stratégies de gestion de branches : branche longue, thématique, ...

## **Travailler avec un dépôt distant**

- Dépôt distant, branches distantes, suivi de branche
- Afficher et inspecter les dépôts distants
- Ajouter, renommer, retirer ses dépôts distants
- Tirer, pousser et supprimer une branche distante

## **Réécrire l'histoire, rebaser**

- Mise en garde : les dangers de la réécriture
- Rebaser une portion de branche
- Quand rebaser et quand fusionner

## **Remiser et nettoyer**

- Remiser son travail en cours
- Créer une branche depuis une remise
- Nettoyer son répertoire de travail

## **Personnaliser Git**

- Configurer l'éditeur par défaut, exclusions automatiques, ...
- Création et utilisation d'alias
- Outils graphiques : Git-Gui, GitKraken, SmartGit, ...
- Créer des filtres : smudge et clean
- Crochets côté client : pre-commit, pre-rebase, post-rewrite...
- Crochets côté serveur : pre-receive, update, post-receive

## **Faire référence à un projet externe**

Principe des sous-modules  
Déclarer, tirer et mettre à jour un sous-module  
Modifier et gérer les conflits sur une bibliothèque externe  
Problèmes des sous-modules

## **Git sur un serveur**

Les protocoles : local, HTTP, SSH, Git  
Création d'un dépôt nu, comptes utilisateurs  
Utilisateur git unique, clés SSH et git-shell  
Démon Git

**Atelier : Mise en place d'un serveur Git**

## **Gestion de dépôt web**

Un serveur simple et léger : GitWeb  
Une plate-forme plus complète : GitLab  
GitLab : configuration des utilisateurs  
GitLab : exploration de projet, suivi des activités, wiki  
GitLab : issue manager, web hooks, revue de code  
Un service hébergé clé-en-main : GitHub  
GitHub : création de compte et configuration  
GitHub : règles de contribution  
GitHub : maintenance d'un projet

**Atelier : Récupération et exploration d'un GitLab**

## **Comprendre l'intégration continue et découvrir GitLab**

Processus de développement, tests unitaires / d'intégration  
Intégration continue : présentation, positionnement dans une démarche agile  
Gestion des environnements : développement, recette, production  
Outils de conteneurs applicatifs (Docker)  
Configurations système et applicative et outils de centralisation (Puppet, Ansible)  
Panorama outils de gestion : versionnement, build, tests, qualité  
GitLab-CI : présentation, fonctionnalités  
Types d'installation  
Notion de projet, documentation (README.md, Wiki, ...)

**Atelier : Mise en place de GitLab, tour d'horizon de l'interface, création de dépôts et paramétrage**

## **Maîtriser les bases du YAML**

YAML : syntaxe de base, spécificités  
Déclaration et utilisation de variables  
Collections  
Ancres

## **Gérer des builds avec GitLab CI**

Principe de fonctionnement : pipelines, stages, tasks, artefacts, tags  
Structure d'un build de projets, le fichier manifeste .gitlab-ci.yml  
Jobs et Runners, utilisation de Docker  
Mise en place de builds : automatiques / manuels  
Plug-ins pour la gestion des dépôts de source  
Outils de build : Maven, Gradle, ...  
Organisation des branches et des tags  
Gestion des dépendances et dépôts, mise en place d'un cache  
Intégration des dépôts avec les outils de build  
Gestion des notifications  
Création et utilisation de variables dans les paramètres CI/CD  
Lancement de jobs en parallèle

**Atelier : Interfaçage avec des dépôts de dépendances - Configuration et lancement de builds (applications web JS ou services Java)**

### **Contrôler la qualité du code**

Présentation, gestion de la qualité du code  
Panorama des outils : Checkstyle, FindBugs, ...  
Rapport de qualité : configuration, plug-ins (Violations)  
Autres rapports : complexité, tâches, ...

**Atelier : Intégration d'outils de gestion de qualité du code (SonarQube) dans une démarche d'intégration continue**

### **Automatiser les tests**

Types de tests  
Automatisation, couverture  
Tests unitaires et d'intégration  
Tests d'acceptance, tests de performances  
Optimisation des tests

**Atelier : Multiples scénarios d'automatisation de tests unitaires, d'intégration, de performances**

### **Mettre en place une stratégie de déploiement**

Stratégie globale d'automatisation  
Scripts de déploiement et de mise à jour  
Rollbacks  
  
Gestion des artefacts (archivage)  
Utilisation des groupes de ressources pour limiter la concurrence

**Atelier : Construction de scripts de déploiement**

### **Administrer les outils**

Sécurité du serveur d'intégration continue  
Gestion des utilisateurs : bases, rôles, autorisations  
Gestion des journaux  
Espace mémoire/charge CPU, espace disque

Monitoring

**Atelier : Multiples tâches d'administration du serveur**