Gestion de versions avec GIT

Référence : **LUUX151** Durée : **2 jours** Certification : **Aucune**

CONNAISSANCES PREALABLES

• Connaissance des processus de développement et d'un langage de programmation.

PROFIL DES STAGIAIRES

• Tout développeur, chef de projet, architecte, souhaitant utiliser git comme gestionnaire de versions.

OBJECTIFS

• Comprendre les principes d'un gestionnaire de version distribué, les apports de git, savoir le mettre en oeuvre pour gérer les codes sources d'un projet, les versions, corrections de bugs, etc.

CERTIFICATION PREPAREE

Aucune

METHODES PEDAGOGIQUES

- Mise à disposition d'un poste de travail par stagiaire
- Remise d'une documentation pédagogique papier ou numérique pendant le stage
- La formation est constituée d'apports théoriques, d'exercices pratiques, de réflexions et de retours d'expérience
- Le suivi de cette formation donne lieu à la signature d'une feuille d'émargement

FORMATEUR

Consultant-Formateur expert Production et Supervision

METHODE D'EVALUATION DES ACQUIS

- Auto-évaluation des acquis par le stagiaire via un questionnaire
- Attestation de fin de stage adressée avec la facture

CONTENU DU COURS

Présentation de Git

- La notion de gestionnaire de versions distribué
- Historique de git, licence
- Présentation des principes techniques de git : architecture, les objets stockés
- Les différentes utilisations de git : utilisation d'applicatifs stockés sous git, développement, partage de codes, gestions de modifications, de correctifs etc.
- Aperçu des types de workflows possibles

Prise en main

- La commande git, options principales
- Installation et configuration de git.Présentation des notions de base : référentiel, index, répertoire de travail, clônage
- Travaux pratiques : Création d'un premier dépôt.Utilisation de la ligne de commande pour les opérations de base ; Enregistrement de modifications simples ; Clônage d'un référentiel existant

Gestion des développement

- Etude des commandes principales de manipulation des fichiers :add, status, diff, commit, ...
- Gestion des branches : branch, checkout, merge, log, stash, etc.
- Travaux pratiques : Mise en oeuvre sur un projet exemple représentatif des principaux cas d'utilisation ; Ajout, modification, suppression de fichiers et répertoires ; Gestion des commits ; Création de branches, navigation entre branches, fusion de branches
- Résolution des conflits
- Intérêt des branches temporaires

Travail collaboratif

- Objectif : partage et mise à jour de projets
- Fonctionnalités requises : mise à disposition des objets, analyse des modifications, intégration,etc.
- Définition des rôles (développeurs, intégrateurs)
- Notion de dépôt local et dépôt centralisé

- Etude des commandes : fetch, pull, push, remote, ...
- Pour le contrôle de fichiers : show, log, diff, ...
- Gestion des patchs : apply, rebase, revert, ...
- Travaux pratiques : Connexion à un réferentiel ; Synchronisation avec un référentiel distant ; Utilisation des tags pour identifier des commits ; Création et application de patchs sur un exemple de projet complet

Administration

- Tâches d'administration : nettoyage des arborescences, vérification de la cohérence de la base de données, état du service git
- Travaux pratiques : Installation d'un dépôt privé centralisé pour une gestion de sources collaborative, import de développements externes avec fast-import

Compléments

- Interagir avec des référentiels partagés via GitHub
- Exemples de projets sur GitHub, GitLab
- Présentation d'outils complémentaires : gerrit, un système de revue de code.Gitweb, l'interface web
- GitKraken, client graphique

Bonnes Pratiques

- Echanges par rapport aux contextes projets et à l'organistion des équipes pour savoir définir l'utilisation de git la plus adaptée à chaque contexte projet