Git - Gestion du contrôle de versions

2 j (14 heures) Ref : GITD

Public

Architectes, Chefs de projets, Consultants, Développeurs, Ingénieurs…

Pré-requis

Savoir pratiquer Java avec Eclipse est recommandé.

Moyens pédagogiques

Formation réalisée en présentiel ou à distance selon la formule retenue
Alternance d'exposés et de cas pratiques, synthèse
Un poste informatique par stagiaire connecté à internet, à une imprimante en réseau et au réseau informatique
Les salles sont équipées d'un tableau interactif ou d'un vidéoprojecteur et d'un paperboard
Support de cours fourni à chaque stagiaire

Modalités de suivi et d'évaluation

Feuille de présence émargée par demi-journée par les stagiaires et le formateur
Exercices de mise en pratique ou quiz de connaissances tout au long de la formation permettant de mesurer la progression des stagiaires
Questionnaire d'évaluation de la satisfaction en fin de stage
Auto-évaluation des acquis de la formation par les stagiaires
Attestation de fin de formation

Jenkins est un outil Open Source de gestion et d'automatisations de tâche. Son usage s'est aujourd'hui orienté vers les processus d'intégration continue pour orchestrer des pipelines de tests et déploiement. Avec une large gamme d'extensions (plus de 1500 plugins), Jenkins s'intègre et agrège de nombreux outils.
A l'issue de la formation, vous serez capable d'installer et de configurer un serveur Jenkins, solution Open Source d'intégration continue.

Objectifs

Connaître les principes de fonctionnement d'un gestionnaire de versions distribué

Découvrir par la pratique la philosophie de Git et ses apports

Créer et initialiser un dépôt avec Git

Manipuler les commandes de Git pour gérer les fichiers et les branches

Mettre en œuvre un projet en mode collaboratif avec Git

Programme détaillé

PLACER GIT DANS LES SYSTEMES DE GESTION DE VERSION

Historique de Git

Git un système de gestion de version décentralisé

Principes de fonctionnement de GIT : snapshots vs per file

Les « états » de GIT (worktree, stage, database)

INSTALLER ET CONFIGURER GIT

Installation de GIT en fonction des plateformes,

Configuration initiale,

Réglages par défaut en fonction des environnements,

Les attributs Git

INITIALISER UN DEPOT GIT LOCAL

Initialiser un dépôt Git local

Cloner un dépôt Git distant

Le principe des branches Git

PRATIQUER GIT AU QUOTIDIEN

Commit, Revert, Reset : gérer les commits,

Rebase, Merge : gérer les fusions,

Stash : éviter des commits inutiles

RE-RE-RE : REuse REcorded REsolution éviter les conflits de fusion récurrents,

Optimiser les commandes avec les raccourcis

COMPRENDRE ET UTILISER L'HISTORIQUE GIT

Log : lire l'histoire des commits

Réécrire les commits

Reflog : la mémoire de Git

Bisect Dissect : résoudre un bug

Tagger

UTILISER LE MODELE DE BRANCHES GIT FLOW

Initialiser git flow dans un projet,

Démarrer, terminer une « feature »

Démarrer, terminer une « release »

Démarrer, terminer un « hotfix »

TRAVAILLER EN EQUIPE AVEC GIT

Créer un dépôt distant sur Github,

Gestion des utilisateurs,

Clone ou Fork : cas d'utilisation,

Fetch, Pull, Push : gérer le dépôt distant,

Cherry-Pick : éviter des back-merges complets,

Pull-requests : revue de code et fusion

UTILISER LES SOUS-MODULES ET LES SOUS-ARBRES

Cas d'utilisation

Travailler avec des SubModules

Travailler avec des Subtrees

UTILISER LES HOOKS POUR CREER UN CI / CD

Présentation des hooks git,

Implémenter un CD avec post-receive