Phirio

Git : Gestion du contrôle de versions

UX151

Durée: 2 jours

1 460 €

20 au 21 janvier
14 au 15 avril

7 au 8 juillet
13 au 14 octobre
18 au 19 décembre

Public :

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

Objectifs :

A l'issue de la formation, le stagiaire sera capable d'installer, de configurer et d'utiliser GIT, solution Open Source de contrôle de versions.

Connaissances préalables nécessaires :

savoir pratiquer Java avec Eclipse est recommandé

Objectifs pédagogiques :

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 oeuvre un projet en mode collaboratif avec Git

Programme :

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

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, ...
Aperçu des types de workflows possibles.

Créer et initialiser un dépôt avec Git

La commande git, options principales
Installation et configuration de git. Présentation des notions de base : référentiel, index, répertoire de travail, clonage

Atelier : Création d'un premier dépôt. Utilisation de la ligne de commande pour les opérations de base. Enregistrement de modifications simples. Clonage d'un référentiel existant.

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

Etude des commandes principales de manipulation des fichiers :add, status, diff, commit, ...
Gestion des branches :branch, checkout, merge, log, stash, ...

Atelier : 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.

Mettre en oeuvre un projet en mode collaboratif avec Git

Objectif : partage et mise à jour de projets.
Fonctionnalités requises : mise à disposition des objets, analyse des modifications, intégration, ...
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, ...

Atelier : Connexion à un référentiel. 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

Atelier : 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'organisation des équipes pour savoir définir l'utilisation de git la plus adaptée à chaque contexte projet.