Smartfuture
Certifications | Training | Technology solutions

Formation

Outils de Contrôle des Versions GIT

Maîtrisez Git et les bonnes pratiques de gestion de versions pour sécuriser, tracer et collaborer efficacement sur vos projets logiciels.

2 jours 14 h	**Format** Atelier pratique
Modalité Présentiel ou en ligne	**Niveau** Initiation à intermédiaire

INSCRIPTION / RÉSERVATION

Je m'inscris maintenant

Formation 100% pratique

Exercices & cas d'usage

Formateur expérimenté

OBJECTIFS PÉDAGOGIQUES

- Comprendre les principes du contrôle de versions distribué et le positionnement de Git.
- Installer, configurer et utiliser Git pour versionner un projet logiciel.
- Maîtriser la gestion des branches, la fusion et la résolution de conflits.
- Collaborer efficacement avec GitHub / GitLab / Azure DevOps (push, pull, merge requests).
- Mettre en œuvre les bonnes pratiques de commit, d'historique et de workflows Git.

PUBLIC CIBLE

- Développeurs et intégrateurs souhaitant structurer leur travail en équipe.
- Chefs de projet, Scrum Masters et responsables techniques impliquées dans des projets agiles.
- Ingénieurs QA, DevOps, administrateurs systèmes amenés à gérer du code ou des scripts.
- Étudiants ou personnes en reconversion vers les métiers du développement.

PRÉREQUIS

- Connaissances de base en développement logiciel (quelque soit le langage).
- Être à l'aise avec l'environnement de travail sur PC et la ligne de commande.
- Une expérience préalable sur un projet logiciel est un plus.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Introduction au contrôle de versions et à Git

Les enjeux du contrôle de versions

- Pourquoi versionner le code ? Historique, traçabilité et collaboration.
- Contrôle de versions centralisé vs distribué : Subversion, TFS, Git.
- Concepts clés : dépôt, révision, branche, tag, merge, conflit.

Découvrir Git

- Principes internes : snapshots, SHA-1, index, local vs distant.
- Panorama des plateformes : GitHub, GitLab, Azure DevOps, Bitbucket.
- Installation de Git sous Windows / Linux / macOS.
- Configuration initiale : identité, éditeur, options globales (git config).

Premiers pas avec un dépôt Git local

Créer et initialiser un dépôt

- Initialisation d'un dépôt (git init) et structure du dossier .git.
- Ajouter des fichiers au suivi : git status, git add, git commit.
- Organisation des fichiers, renommage et suppression (git mv, git rm).

Comprendre le cycle de vie des fichiers

- États d'un fichier : untracked, staged, tracked, modified.
- Utilisation de .gitignore pour exclure des fichiers.
- Consulter l'historique : git log, options de visualisation.
- Comparer les versions : git diff, comparaison index / HEAD / working tree.

Branches, fusion et résolution de conflits

Gestion des branches

- Créer et lister des branches (git branch).
- Changer de branche (git checkout / git switch).
- Stratégies de branches : main / develop, branches de fonctionnalités, de release, de hotfix.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Fusionner des branches

- Comprendre la fusion fast-forward et non fast-forward.
- Fusion simple (git merge) et gestion des commits de merge.
- Visualiser le graphe de commits pour comprendre l'historique.

Résolution de conflits

- Identifier les conflits survenus lors d'une fusion.
- Résoudre un conflit dans un éditeur (VS Code, outils graphiques Git).
- Valider la résolution (git add, git commit) et bonnes pratiques..

Travailler avec des dépôts distants

Cloner et connecter un dépôt distant

- Créer un compte GitHub / GitLab / Azure DevOps.
- Cloner un projet existant (git clone) et notion de remote.
- Configurer plusieurs dépôts distants (origin, upstream, etc.).

Synchroniser son travail

- Récupérer les changements (git fetch, git pull).
- Envoyer ses commits (git push, suivi des branches distantes).
- Gestion des droits d'accès et des clés SSH.

Collaboration et revues de code

- Fonctionnement des pull requests / merge requests.
- Bonnes pratiques de revue de code et d'intégration continue.
- Exemples de workflows : GitHub Flow, GitLab Flow, Git Flow (présentation).

Historique avancé, réécriture et dépannage

Explorer et analyser l'historique

- Options avancées de git log (graph, oneline, decorate, filtres).
- Identifier l'origine d'un bug avec git blame.
- git reflog : retrouver des commits perdus.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Modifier. et corriger l'historique

- Corriger le dernier commit (git commit --amend).
- Annuler des changements avec git revert.
- Comprendre git reset (soft, mixed, hard) et ses impacts.
- Rejouer des commits sur une autre branche (git cherry-pick).

Mettre de côté et reprendre du travail

- Utiliser git stash pour suspendre temporairement des modifications.
- Lister, restaurer et supprimer des stash

Bonnes pratiques Git & intégration avec l'environnement de développement

Conventions, qualité et automatisation

- Rédiger de bons messages de commit (style, structure, références tickets).
- Structurer un dépôt : organisation des dossiers, README, licence.
- Tags et versions : marquer les releases et livrables.
- Hooks Git : automatiser des contrôles (lint, tests) avant commit/push.

Intégration avec les IDE et la CI/CD

- Utiliser Git dans Visual Studio Code ou d'autres IDE (vue graphique, staging, diff).
- Lien entre Git et les pipelines CI/CD (présentation générale).
- Exemples de bonnes pratiques d'équipe pour la livraison continue.

Atelier pratique fil rouge

Mise en situation complète

- Initialisation d'un nouveau projet Git et configuration du dépôt distant.
- Travail collaboratif : création de branches de fonctionnalités, pull requests, revues de code.
- Simulation de corrections de bugs urgents (hotfix) sur une version en production.
- Mise en place d'une stratégie de branches adaptée au contexte de l'entreprise